A Modular Classifier Concept for Activity Recognition on Mobile Phones

Ein modulares Konzept von Klassifikatoren für Aktivitätserkennung auf Mobiltelefonen

Von der Carl-Friedrich-Gauß-Fakultät
der Technischen Universität Carolo-Wilhelmina zu Braunschweig

zur Erlangung des Grades

Doktor-Ingenieur (Dr.-Ing.)

genehmigte

DISSERTATION

von **Martin Berchtold-Buschle**

Eingereicht am:	17. August 2011
Mündliche Prüfung am:	26. Oktober 2011
Druckjahr:	2012
1. Referent:	Prof. Dr. Lars Wolf
2. Referent:	Prof. Dr. Michael Beigl

Zusammenfassung

Aktivitätserkennung ist seit vielen Jahren ein Forschungsgebiet. Mit Aktivitätserkennung wird die Aktivität von Benutzern mithilfe von Sensoren, die am Körper getragen werden, festgestellt. Geeignete Sensoren für die Erkennung sind Mikrofone, Gyroskope, GPS-Empfänger und insbesondere Beschleunigungssensoren. Die Messungen der Sensoren werden digital interpretiert, Merkmale dann extrahiert und entsprechend dieser Merkmale wird eine Klassifikation durchgeführt. Wird ein Beschleunigungssensor für die Aktivitätserkennung eingesetzt, dann können generelle Aktivitäten mit viel Bewegung, wie z.B. *Gehen*, *Treppen steigen* und *Fahrradfahren*, und ohne Bewegung, wie z.B. *Sitzen*, *Stehen* und *Liegen*, erkannt werden.

Es gibt viele klassische Anwendungsgebiete für Aktivitätserkennung wie z.B. *Altenpflege*, *Gesundheitspflege*, *Fitness*, *Sport*, *Beobachtung von Arbeitsabläufen* und *Gedächtnisunterstützung*. In der *Altenpflege* wird die Aktivitätserkennung z.B. zur automatischen Feststellung von Stürzen benutzt, um dann direkt den zuständigen Pfleger zu alarmieren. Ein weiteres Beispiel für Aktivitätserkennung in der *Altenpflege* ist das Überwachen von Aktivitäten, damit der Fortschritt von Krankheiten wie z.B. Alzheimer oder Demenz festgestellt werden kann. Für *Sport* könnten z.B. spezielle Bewegungen erkannt werden, damit falsches Trainingsverhalten festgestellt oder der Kalorienverbrauch berechnet werden kann.

Viele weitere Beispiele für Applikationen, die Aktivitätserkennung nutzen, existieren, wobei üblicherweise für die Erkennung spezielle Hardware, die in einem Körpernetzwerk angeordnet ist, verwendet wird. Diese Hardware muss für diesen Zweck speziell angeschafft, angebracht und gewartet werden, aber für allgemeine Benutzer, welche kein Wissen über Sensornetzwerke besitzen, ist die Verwendung solcher Systeme nicht praktikabel. Da die meisten Menschen in westlichen Ländern Mobiltelefone besitzen, welche mit Sensorik ausgestattet sind, würde die Benutzung von Aktivitätserkennung diese in die Lage versetzen, künstliches Bewusstsein in ihr Leben zu integrieren. Der Benutzer muss sich einfach die Algorithmen für die Aktivitätserkennung herunterladen und sich nicht erst spezielle Hardware anschaffen.

Zusätzlich zu den klassischen Feldern der Aktivitätserkennung kann das spezielle Gerät, wie es ein Mobiltelefon ist, selbst von der Erkennung profitieren. Da das Mobiltelefon meist beim Benutzer, nahezu niemals ausgeschaltet und sehr häufig online ist, können verschiedenste peinliche, stressige und selbst gefährliche Situationen aufgrund von Unterbrechungen durch Mobiltelefone entstehen. Mit der mobilen Internetbenutzung können diese Situationen potenziell noch häufiger auftreten. Hier kann die Aktivitätserkennung auf dem Mobiltelefon das Gerät wieder unaufdringlich machen und im Hintergrund verschwinden lassen, bis wirklich eine Interaktion mit ihm gewünscht ist.

So wünschenswert die Aktivitätserkennung auf und mit Mobiltelefonen ist, so viele Herausforderungen bestehen mit dieser Art Gerät. Fünf generelle Herausforderungen konnten durch die Forschung mit Mobiltelefonen identifiziert werden:

1. **Flexibilität:** Die Gegebenheiten der Benutzung des Mobiltelefons und deshalb der Aktivitätserkennung können sich jederzeit ändern. Die mentalen oder physischen Bedingungen des Nutzers können variieren, die Kleidung oder das Verhalten sich ändern oder der Benutzer selbst wechseln. Eine Aktivitätserkennung muss sich **flexibel** an diese Veränderungen anpassen lassen, so dass die Erkennung verlässlich für den individuellen Benutzer funktioniert.
2. **Erweiterbarkeit:** Verschiedene Benutzer haben verschiedene Ansprüche an Aktivitäten die erkannt werden sollen. Auch verschiedene Applikationen benötigen die Erkennung von verschiedenen Aktivitäten. Nur eine kleine Menge von Aktivitäten wird von nahezu jedem Benutzer durchgeführt. Deshalb muss die Erkennung **erweiterbar** für die individuellen Bedürfnisse sein.
3. **Robustheit:** Da das Mobiltelefon überall am Benutzer oder in seiner Umgebung sein kann, sind die Muster die die Sensoren messen sehr unterschiedlich. Auch ist das Gerät nicht fest an einem Ort angebracht, was in sehr verrauschten Daten resultiert. Eine **robuste** Erkennung wird benötigt, welche trotz der verrauschten und variablen Muster der Sensoren fähig ist, die Aktivitäten mit einer hohen Genauigkeit zu erkennen.
4. **Ressourcen:** Die Ressourcen auf Mobiltelefonen sind stark limitiert. In diesem Sinne sollte die Last des Prozessors durch die Aktivitätserkennung nicht die normale Telefonbenutzung beeinträchtigen. Auch sollte die Laufzeit der Batterie nicht stark eingeschänkt werden.
5. **Konditionalität:** Der Benutzer und sein Telefon können in verschiedensten Gegebenheiten situiert sein. Jede dieser **Konditionen** impliziert andere Muster der Sensoren, welche jeweils durch den Algorithmus der Aktivitätserkennung repräsentiert sein müssen. Eine Aktivitätserkennung für Mobiltelefone muss auf diese verschiedenen **Konditionen** reagieren können, wobei die Komplexität weder erheblich gesteigert noch die Genauigkeit stark verringert werden darf.

In dieser Dissertation wird eine modulare Aktivitätserkennung mit Beschleunigungssensoren auf Mobiltelefonen vorgestellt, welche Lösungen für alle fünf Herausforderungen beinhaltet. Durch die Modularität wird ermöglicht, dass gewisse Teile der Aktivitätserkennung angepasst werden und dadurch **Flexibilität** bereitgestellt werden kann. Eine modulare Erkennung ist **erweiterbar** durch neue Module, welche neue Aktivitäten erkennen. Die Rekurrenz des Klassifikationsprozesses stabilisiert die Erkennung und ermöglicht die Ableitung eines Zuverlässigkeitsmaßes. Gemeinsam resultitert dies in einer **robusten** Aktivitätserkennung. Nur ein Modul und nicht die ganze Aktivitätserkennung ist zu einem Zeitpunkt aktiv, was den Berechnungsaufwand reduziert, so dass **Ressourcen** gespart werden können. Jedes Modul kann passend sein, um mit einer **Konditionalität** umzugehen, wobei die Komplexität weder erheblich erhöht noch die Genauigkeit stark erniedrigt wird. Alle diese Lösungen für die Herausforderungen der Aktivitätserkennung auf Mobiltelefonen werden durch einen Service abgerundet, welcher das neuartige System auf den Mobiltelefonen normaler Benutzer unterstützt.

Abstract

Activity recognition has been a research topic for many years now. With activity recognition the users activity is detected based on sensors worn on the body. Suitable sensors for recognition are microphones, gyroscopes, GPS receivers and especially accelerometers. Measurements from these sensors are digitally interpreted, features are then extracted and according to these features a classification can be made. When an accelerometer is used for activity recognition, general activities of the user entailing movement, e.g. *walking*, *climbing stairs* and *cycling*, and without movement, e.g. *sitting*, *standing* and *lying*, can be detected.

There are various classical application fields for activity recognition such as *elderly care*, *health care*, *fitness*, *sports*, *workflow monitoring* and *memory support*. In *elderly care*, activity recognition is e.g. used to detect the accidental activity *falling down*, in order to automatically alert the responsible caregiver. Another example of activity recognition for *elderly care* is the monitoring of the activities, so disease progression of e.g. Alzheimer's or dementia can be detected. In the field of *sports*, e.g. special movements can be recognized, so that incorrect training behavior can be detected or calorie expenditure due to sports can be calculated.

There are many more examples of applications for classical activity recognition, which is usually done with special hardware arranged in body area networks. This hardware needed to be acquired, deployed and maintained, but for common users not skilled in sensor networks the utilization of such sensors is not feasible. Since most people in western countries have mobile phones, which are equipped with sensors for pattern recognition, the utilization for activity recognition would enable the common user to include this kind of artificial awareness in her life. The user just has to download an algorithm for activity recognition and does not need to buy additional hardware.

In addition to the classical application fields of activity recognition, a mobile phone device can benefit from the recognition itself. Since the mobile phone is mostly with the user, nearly never switched off and very often online, various embarrassing, stressful or even dangerous situations are possible due to interruptions by the mobile phone. With mobile Internet usage these situations are potentially even increasing in frequency. Here the activity recognition on the mobile phone can make it unobtrusive again and make it disappear into the background until it is actually needed.

As desirable as the activity recognition on and with mobile phones is, there are a lot of challenges with this kind of device. Five general challenges could be identified due to the research with mobile phones:

1. **Flexibility:** The conditions of the mobile phone usage and therefore for the activity recognition can always change. The users physical or mental condition can vary, the clothing and behavior can change or even the user herself. An activity recognition needs to **flexibly** adapt to this changes, so that the recognition works with high accuracy for the individual user.
2. **Extensibility:** Different users have different demands of activities to be recognized. Also, different application fields need the recognition of different activities. Only a small set of activities are performed by nearly every user. Therefore, the recognition needs to be **extensible** to the individual needs.
3. **Robustness:** Since a mobile phone can be everywhere on the user or her environment, the sensor patterns vary. Also, the device is typically not firmly attached to any position, which results in noisy sensor data. A **robust** recognition is needed, which despite the noisy and varying sensor patterns is able to detect the activities with high accuracy.
4. **Resources:** The resources on mobile phones are limited. In this manner the processor load due to the activity recognition should not affect the normal phone usage. Also, the battery runtime should not significantly be reduced due to the activity recognition.
5. **Conditionality:** The user and her phone can be situated in various different **conditions**. Each of these **conditionalities** implies different sensor patterns, which need representation in the activity recognition algorithm. An activity recognition for mobile phones needs to react onto these different **conditionalities** without significantly increasing in complexity or decreasing in accuracy.

In this thesis a modular activity recognition using accelerometer sensors on mobile phones is presented, which includes solutions to all five challenges. The modularity enables the individual adaption of parts of the activity recognition to offer **flexibility**. A modular recognition is **extensible** by new modules which detect new activities. The recurrence of the classification process stabilizes the recognition and enables the derivation of a reliability measure, which in conjunction result in a **robust** activity recognition. Only one module and not the whole activity recognition is active at each point in time, which decreases the calculation effort and therefore the energy consumption, which saves **resources** in general. Each module can be suited for dealing with one **conditionality**, through which neither the complexity of the recognition is increased nor the accuracy is significantly lowered. All these solutions to the challenges of activity recognition on mobile phones are rounded by a service, which supports the novel system on the common user's phone.

Danksagung

Diese Dissertation geht aus meiner langjährigen Arbeit in dem Gebiet der Aktivitätserkennung hervor. Die erste Inspiration mich in diesem Gebiet zu engagieren kam bei meiner Studienarbeit bei Kristof van Laerhoven an der Universität von Lancaster auf. Ihm gebührt besonderer Dank, da er mich nicht nur an das Gebiet herangeführt hat, sondern mir auch immer wieder mit Rat und Tat zur Seite stand.

Viele meiner Ideen sind aus meiner langjährigen Arbeit als Hilfswissenschaftler und wissenschaftlicher Mitarbeiter am TECO des Karlsruher Instituts der Technologie (KIT) hervorgegangen. Besonders die fruchtbaren Diskussionen und die Unterstützung durch meine Kollegen Till Riedel und Christian Decker haben meine Arbeit vorangebracht. Deren manchmal kritische Kommentare haben mir sehr geholfen die Schwachpunkte meiner Theorien zu offenbaren und zu beseitigen, weshalb ihnen mein besonderer Dank gilt. Die Ursprünge vieler Themen die ich in dieser Dissertation behandelt habe gehen auch auf meine Diplomarbeit zurück, deren Themengeber und Betreuer Tobias Zimmer ich danken möchte.

Nach meinem Wechsel an die TU Braunschweig zu der Gruppe von Michael Beigl konnte ich meine Forschung auf ein neues Anwendungsgebiet ausrichten. Michael Beigl ist nicht nur mein Doktorvater, sondern hat mir auch maßgeblich geholfen dieses Anwendungsgebiet zu identifizieren. Er hat mich auch während meiner Tätigkeit am Institut für Betriebssysteme und Rechnerverbund (IBR) immer wieder motiviert meine Arbeiten voran zu treiben und zu publizieren, weshalb ich ihm zu großem Dank verpflichtet bin. Auch meinem Kollegen am IBR Dawud Gordon möchte ich hier danken, denn gerade seine Tätigkeiten im Bereich Aktivitätserkennung haben mir gegen Ende meiner Forschungen für meine Dissertation die Grundlage für viele hilfreiche Diskussionen geliefert.

Die Forschungen innerhalb des Projekts IT-Ökosysteme des Landes Niedersachsen halfen mir mich wissenschaftlich frei zu entfalten und haben letztendlich dazu beigetragen meine Dissertation zu vollenden. In dieser Hinsicht gilt auch großer Dank dem Teilprojektleiter und letztendlichem Referenten meiner Dissertation Lars Wolf, der mich auch dann noch betreut hat nachdem ich teilweise wieder am KIT beschäftigt war.

Auch ich hatte Unterstützung durch Studenten während meiner Tätigkeit als wissenschaftlicher Mitarbeiter und Arbeit an meiner Forschung. Danken möchte ich den studentischen Unterstützern, allen voran Henning Günther, der viele meiner mehr theoretischen Arbeiten in die Praxis umgesetzt hat, Firas el Simrany, der für etliche meiner Publikationen das lästige Sammeln von Sensordaten übernommen hat, und Jan-Hendrik Hanne, der sich um die Implementierungen auf der doch sehr speziellen Plattform iPhone gekümmert hat. Sehr großer Dank gebührt auch meinem Freund und ehemaligen Kollegen Matthias Budde, der sich jedem noch so wirren meiner Texte angenommen hat und versucht hat ihn für Dritte verständlich zu machen.

Bei meinen Eltern Paula und Anton, meiner Familie und speziell Hannelore und Werner Buschle möchte ich mich auch bedanken, denn sie haben mich ermutigt und mich sehr dabei unterstützt meine Promotion durchzuführen. Zu guter Letzt möchte ich meiner Freundin Nagihan Kücükyildiz danken, die mich gerade in schweren Zeiten mit viel Verständnis und Liebe wieder aufgebaut hat.

Contents

1 Introduction **10**
1.1 Rationale . . . 12
1.1.1 Calm Mobile Phone . . . 12
1.1.2 Classical Applications for Activity Recognition . . . 12
1.2 Problem Statement . . . 13
1.2.1 Challenge 1 - Flexibility . . . 13
1.2.2 Challenge 2 - Extensibility . . . 14
1.2.3 Challenge 3 - Robustness . . . 14
1.2.4 Challenge 4 - Resources . . . 14
1.2.5 Challenge 5 - Conditionality . . . 15
1.3 Scope and Methodology . . . 15
1.4 Contribution . . . 15
1.5 Outline of the Thesis . . . 16
1.6 Publications . . . 17

2 Basic Principles and Related Work **18**
2.1 Activity and Context Recognition . . . 20
2.1.1 Elderly Care . . . 20
2.1.2 Fitness, Well-being and Health Care . . . 21
2.1.3 Psychology . . . 23
2.1.4 Workflow Monitoring . . . 24
2.1.5 Groupware and Memory Support . . . 25
2.1.6 Miscellaneous or not bound to any Application Field . . . 25
2.1.7 Summary and Overview . . . 27
2.2 Sensor Data Processing . . . 29
2.2.1 Feature Extractions . . . 29
2.2.2 Mapping Functions and Classifiers . . . 33
2.3 Modular Classification . . . 39
2.3.1 Soft-Output Classifier Fusion Methods . . . 39
2.3.2 Class Ranking Based Techniques . . . 40
2.3.3 Fusing Single Class Labels . . . 40
2.3.4 Methods Operating on Classifiers . . . 41
2.3.5 Summary and Discussion . . . 41

3 Modular Activity Recognition Architecture **44**
3.1 The Processing Queue . . . 46
3.1.1 Accelerometer Sensor Sampling . . . 46
3.1.2 Mean and Variance Feature Extraction . . . 47
3.1.3 Recurrent Fuzzy Inference System (RFIS) Mapping Function . . . 49
3.1.4 Fuzzy Classification . . . 51
3.1.5 Reliability Filter . . . 53
3.1.6 Bit-Vector Masking . . . 55
3.2 Modular Classification . . . 58
3.2.1 Classification Module . . . 58
3.2.2 Dynamic Queue of Classification Modules . . . 60
3.2.3 Probability of Module Transitions . . . 61
3.2.4 Other Classifier Module Scheduling Techniques . . . 63
3.3 Modular Classifier Training Algorithm . . . 65
3.3.1 Data Grouping . . . 65
3.3.2 Subtractive Clustering . . . 66
3.3.3 Gath-Geva Clustering . . . 68

3.3.4 Linear Regression . . . 69
3.3.5 Genetic Algorithm . . . 71
3.3.6 Training Algorithm . . . 73
3.3.7 Bit-Vector Search . . . 75
3.4 Tools Supporting the Modular Classifier Architecture and Implementations . . . 77
3.4.1 Implementations for Different Phones . . . 77
3.4.2 Classifier Specification Format (CSF) . . . 80
3.4.3 Training Algorithm Implementations . . . 82
3.4.4 Context Annotator Tool (CAT) . . . 84
3.4.5 Data Collector Tool (DCT) . . . 87
3.4.6 Annotation Package Format (APF) . . . 88

4 Challenges **92**
4.1 Challenge 1 - Flexibility . . . 94
4.1.1 Exchange . . . 94
4.1.2 Adaptation . . . 95
4.2 Challenge 2 - Extensibility . . . 96
4.2.1 Problem Statement . . . 96
4.2.2 Solution . . . 97
4.3 Challenge 3 - Robustness . . . 100
4.3.1 Methods Deriving Reliability Measures . . . 100
4.3.2 Robust Recurrent Classification . . . 102
4.4 Challenge 4 - Resources . . . 104
4.4.1 Processor Load . . . 104
4.4.2 Energy Consumption . . . 107
4.5 Challenge 5 - Conditionality . . . 110
4.5.1 Conditional Context . . . 110
4.5.2 Phone Orientation . . . 111
4.6 Tradeoffs . . . 113
4.6.1 Activity Classification Accuracy vs. Activity Event Detection Rate . . . 113
4.6.2 Robustness vs. Resources . . . 114

5 Evaluation **116**
5.1 Evaluation: Challenge 1 - Flexibility . . . 118
5.1.1 Bit-Vector Adaptation . . . 118
5.1.2 Adaptation and Exchange of Modules in a Dynamic Queue . . . 120
5.2 Evaluation: Challenge 2 - Extensibility . . . 129
5.2.1 Evaluation . . . 129
5.3 Evaluation: Challenge 3 - Robustness . . . 134
5.3.1 Fuzzy Classifiers vs. Recurrent Fuzzy Classifiers . . . 134
5.3.2 Filtered vs. Non-Filtered RFIS Classification . . . 135
5.4 Evaluation: Challenge 4 - Resources . . . 139
5.4.1 Initial Experiments . . . 139
5.4.2 Complete Analysis of Dynamic Queue Complexity . . . 143
5.4.3 Energy Consumption - OpenMoko Freerunner . . . 146
5.5 Evaluation: Challenge 5 - Conditionality . . . 150
5.5.1 Evaluation . . . 150
5.6 Evaluation: Tradeoffs . . . 155
5.6.1 Activity Classification Accuracy vs. Activity Event Detection Rate . . . 155
5.6.2 Robustness vs. Resources . . . 156

6 ActiServ - An Activity Recognition Service for Mobile Phones 160
6.1 The Architecture . . . 162
6.1.1 General Overview of the Service Components . . . 162
6.1.2 Workflow . . . 165
6.1.3 Global Trainer Service (GTS) . . . 167
6.1.4 Personal Trainer Service (PTS) . . . 169
6.2 Offline Activity Classification Evaluation . . . 172
6.2.1 Evaluation Setting . . . 172
6.2.2 ACMS Trained on Evaluation User . . . 173
6.2.3 Evaluation User Excluded from Training . . . 174
6.2.4 Summary and Discussion of Results . . . 176
6.3 Summary and Conclusion . . . 176

7 Conclusions 178
7.1 Summary . . . 178
7.2 Contribution . . . 179
7.3 Future Work . . . 180

1 Introduction

In the 1980's the first commercial mobile phones arose. Since then the market of mobile communication has developed rapidly. In the early days only a few people had mobile phones, which were large in size and their only purpose was to make calls. Nowadays nearly everybody in industrial countries has at least one cell phone. Todays mobile phones are mostly equipped with processors and memory which only a few years earlier have been limited to normal desktop or laptop computers. Late developments include clock rates in the giga-Hertz range, multi core processors and the bandwidth for communication rose accordingly. The mobile Internet is now accessible from nearly any mobile phone with a speed, that was formerly limited to wire-based high speed DSL or cable. Therefore, the mobile phone and in particular smart phones are more than devices to just make calls.

With devices that are constantly with the user and are mostly online, the user interacts more often. In contrast to a desktop or even a laptop computer, the mobile phone is normally never switched off. Therefore, applications can always interrupt the user and these intermissions could cause undesired situations. For the user this means more stress, not ideal settings or they even can end up in hazardous situations. Accordingly, the effort a user has to invest in this mobile computer somehow has to be reduced to situations where the user explicitly operates an application on the phone.

A method to introduce awareness to mobile phones and to interact with them implicitly due to normal behavior is context recognition. In this manner the *context* concerns all the information about the user and her environment to make computing devices unobtrusive again, so they disappear in the background. A special subset of *context* is *activity*, which encloses more information about the user's current state and not so much about the environment. This *activity* information is exactly what is needed to implicitly control mobile devices, since the single user is the information source, not the surroundings or other people. An example of such activities is given in figure 1.

Figure 1: Example of activities being recognized with a mobile phone.

There are different methods of delivering information about the user's activity, where the sensor-based single user activity recognition is the most relevant for enabling mobile phone awareness. In this case sensors supply measurements from the environment, where different sensors can provide different information about the real world. This measurements need somehow to be processed, so an information about the users activity can be derived. To avoid the necessity of an expert supplying a model or pre-parameterized algorithm to derive activity information

from sensor data, at some point machine learning is used. The machine learning is used to teach an activity recognition system which sensor patterns indicate what activity.

In the course of mobile navigation, multi media, human computer interfaces (HCI) and games, sensors were integrated into mobile devices. A very early example is the project TEA [125], where a Nokia mobile phone was equipped with an external sensor board. There are many sensors which can be utilized with mobile phones, such as microphones, GPS sensors, cameras, etc. The most challenging but also the most expressive sensors for activity recognition are accelerometers.

In this thesis a novel modular activity recognition especially suited for mobile phones is introduced. This modular system can face the challenges of activity recognition with accelerometer sensors on mobile phones, not only to make the devices unobtrusive again, but also to enable the most common sensing device to run activity awareness applications.

First, in this introduction the activity recognition is further motivated. The utilization of activity information is subdivided into two main branches: one to make the phone as unobtrusive as possible and the other one encloses all classical applications for activity recognition. A phone which is not disturbing the user or causing embarrassing, stressful or potential hazardous situations is called *calm mobile phone*. Since most of the classical applications for activity recognition utilize special hardware, the projection of these applications onto mobile phones would make them available to the common user.

Second, the problems of activity recognition with accelerometers on mobile phones are described along five challenges. These challenges were refined during the work with mobile phones and reflect the specialties of sensing with that platform. No standardization or definition of such challenges could be found throughout the literature, so the definition was derived according to experience. Each challenge reflects a special aspect, but the challenges are not independent from each other.

Third, the scope and methodology of this thesis are defined. The scope is the activity recognition with accelerometer sensors on mobile phones. To show the capabilities of this approach the methodology is aligned with the previously defined challenges.

Fourth, the contribution to the field of activity recognition is presented, which is the novel modular classification process along with other improvements. All the contributions are listed with respect to the challenge it helps to cope with. A service to support the modular activity recognition on the user's mobile phone is rounding up the set of contributions.

Sixth, an outline is given, describing in which part of the thesis what part of the architecture is analyzed, evaluated or discussed.

Last, in this introduction all the papers previously published which include parts of this thesis are listed.

1.1 Rationale

There are two main reasons for introducing activity recognition to mobile phones. Since the mobile phone is always with the user and mostly online, there are several occasions where the interaction with the device is disturbing, embarrassing or even dangerous. This causes stress, therefore the utilization of activity information can make this device as unobtrusive as possible.

Activity recognition has been a research topic for many years now in ubiquitous, pervasive and mobile computing. There are several classical application fields like *elderly care*, *health care*, *workflow monitoring* and *fitness*. Usually the recognition in these fields is based on small sensing devices firmly attached to various body positions, but with mobile phones having the sensors built in a new, nearly always available sensing platform arises.

1.1.1 Calm Mobile Phone

The general term *calm computer*, which describes devices and computers which are invisible and quiet servants, was coined by Mark Weiser [150]. This term fits perfectly the desire of making mobile phones unobtrusive and preventing the user to end up in embarrassing, disturbing or hazardous situations. Therefore, the term *calm mobile phone* is created here, which describes the phone not being center of attention all the time, but disappearing in the background until needed.

Mobile phones are usually with the user or somewhere not far away in their environment. As the user can be situated in various situations, a phone ring might be disturbing e.g. when the user is in a meeting, a ringing cell phone might interrupt a presentation. If the volume is turned off manually, but the vibration alarm is still switched on, the same embarrassing situation can occur, if the phone is lying on a table. Another example is, when the user is receiving a call while riding a bike. The user might pick up the phone while still driving, which is taking all of her attention. Due to this lack of attention, possibly dangerous situations could arise and an accident might occur. Activity recognition could solve this problem. In the meeting situation the phone would first recognize the phone being in the users pants pocket when the user is sitting. An automatic profile adjustment according to the activity recognition might, in this case switch off the ring tone, but activate the vibration alarm. Should the phone be lying on the table, both the ring tone and the vibration could be turned off, leaving only the visual call indication. For the situation where the user is riding a bike, all incoming calls would be rejected and the user notified if she is changing her activity. These two situations were published in a video [24] as argument for activity recognition on mobile phones, but there are many more possibilities.

Especially due to the widely spread mobile Internet usage new situations can occur where the phone needs explicit interaction. E.g. when instant messenger software is running on the phone, the user can always be online and therefor possibly always be reachable. If the user is in a club dancing, but her friend wants to join, the user might not notice incoming messages. The friend might try for a while, but with increasing time of no response, the friend might get angry. In this case a misunderstanding can occur, where the friend is disappointed, but the user just has not noticed the messages. Here also activity recognition can help preventing this misunderstanding by setting the instant messenger status to *busy* and specifying a message for the possible respondent that includes the current activity.

The list of possible applications benefiting from activity recognition is extensive. Also, with web 2.0 applications the user can always be online for the community to be reached, more embarrassing, disturbing, or dangerous situations can happen. As more new application fields for mobile Internet usage arise, the more relevant it is to have the phone react to the different situations and especially the activities the user performs.

1.1.2 Classical Applications for Activity Recognition

Activity recognition for various classical application fields was and is mostly done with small sensing devices distributed and firmly attached to several body parts of the user. In real life it is hardly practical, that e.g. an elder person is equipped with such devices to detect falls or monitor disease progression. With mobile phones being equipped with sensors, especially accelerometers, the detection of the activities can be done on the phone. Since most people nowadays already have mobile phones which are mostly on the person, the possibility of downloading an application for activity recognition is much higher than buying new hardware. Also, this hardware needs to be worn all the time, because what is a system for fall detection doing any good if the user is not wearing it whilst he or she falls.

Another issue with small specialized sensing hardware is, that it needs to be maintained. The devices need some kind of energy source, which is mostly a battery, but batteries run out of energy, so they need to be replaced. With every sensing device having its own battery, each of the power sources need to be maintained. Furthermore, these devices can individually reach faulty software states, in which they are not functioning any more. In this cases they might need to be reset, but the user first needs to recognize this. If a mobile phone is in a faulty state, the user can recognize this whenever she tries to operate it again. For small sensing devices mostly without display or any purpose other than for activity recognition, the detection of such faulty states could just be coincidental and take a long time.

In this manner the mobile phone is the perfect sensing platform for classical application fields of activity recognition, which most people already have. The devices not only include many sensors, but also offer strong processing capabilities. Over the years there even will be more and more senors included in the phones and the processing capabilities will rise. Furthermore, other accessories such as watches [103], headsets [141] or shoes [123] could be equipped with sensors and included in the activity recognition on the mobile phone, but this is clearly future work and out of the focus of this thesis.

All these reasons argue for using mobile phones to recognize activities also for classical applications of the fields like e.g. *sports*, *fitness*, *psychology*, *elderly* and *health care*. For example the daily calorie expenditure [99] of the user could automatically be determined on basis of activity recognition on mobile phones and the user can then directly be informed. Combining the knowledge about the daily activities with information about the food that was eaten, the weight gain or loss can directly be calculated. Furthermore, the progression of a disease such as Alzheimer's or dementia could be monitored and then directly be reacted upon by the caregiver. Many more classical applications could be transfered or implemented on mobile phones. Due to the networking capabilities of mobile phones, the activity recognition can be externally maintained or the informations detected directly used to inform the responsible people.

1.2 Problem Statement

As desirable as it is to do activity recognition with internal accelerometers on mobile phones, there are a lot of challenges arising with this device and type of sensor. The mobile phone is usually not firmly attached to any body position, unless the user carries it on the belt. This results in very noisy and varying sensor data patterns. Also, the phone can be anywhere in the environment or on the user. In each position different activities can be recognized with varying reliability. Furthermore, the condition of the user or the environment can change at any time, which results again in different sensor patterns. The mobile phone has limited resources in calculation and battery capacity. Different users or applications have also different requests of which activities should be recognized. All these characteristics to activity recognition with mobile phones led to the following five challenges.

1.2.1 Challenge 1 - Flexibility

There are always conditions of the user or the environment that can change, e.g. clothes, physical or mental condition, behavior, interests or even the users themselves. Was the activity recognition built to recognize activities with the phone in the pants pocket for a tight trouser, the recognition might not work well with more loose baggy pants. The user could also be suffering from a temporary physical condition like a tremor due to stress. This would certainly influence the recognition for activities where the phone is being held by the user. Furthermore, physical or mental exhaustion leads to different behavior and therefor to different sensor patterns. If the activity recognition system is not constructed according to these patterns, the detection of activities might be faulty. People also change in general and behave differently over time. These changes are very slow and differences could occur over a time span of years. The last example is that the user of a cell phone can change, since it can be sold to a new user.

Since all these changes can not be anticipated during design time of the activity recognition, the system needs to be **flexible** and adapt to changes when they occur. Also, if all these conditions are known at design time, which is nearly impossible due to the high amount of degrees of freedom, the activity recognition system could be designed to cope with them all. This would lead to a large system, which would not be able to cope with the other challenges.

1.2.2 Challenge 2 - Extensibility

Every user has individual interests in sports, leisure time and work. One could be very interested in playing football, the other one more so in hiking and the third has no interest in sports at all. Some people need their work to be part of the activity recognition, but there are various jobs with totally different activities. A small set of activities is nearly performed by everybody, e.g. every person who is not handicapped will occasionally be walking.

Also, different applications need the recognition of different activities. E.g. an application for elderly care, which performs emergency calls when an elder person falls, definitely needs the recognition of the accidental activity of falling down. Furthermore if the employer recommends the activity recognition for workflow monitoring to detect faults or accidents, the activities of that job needs to be detected.

An activity recognition which should detect all these activities for all the different individuals would be huge. Several hundred activities would need to be detected by such a system. This would lead to system which is huge and low in accuracy, which conflicts with the other challenges of activity recognition. Therefore, an activity recognition needs to be **extensible** in addition to the standard activities which every user performs.

1.2.3 Challenge 3 - Robustness

As mentioned before, there is a significant difference between classical activity recognition with small firmly attached sensing devices and the recognition with mobile phones. The phone is usually not firmly attached to any body position, in contrast to on body sensor networks. If, for instance, the mobile phone is carried in the pants pocket, the device can move around which produces very noisy sensor patterns. Also, the orientation of the device can shift over time.

Another aspect is, that due to the different positions and orientations the phone can be carried in, some activities can be recognized more precisely than others. E.g. if the phone is lying on a table and this should be recognized, these patterns are clearly distinguishable from others with much more movement. But, if the activities of *sitting* and *holding the phone* should also be recognized, which are much more similar to the phone *lying on the table*, the recognition accuracy for these activities will be low. Although, not all classifications are equally unreliable, since if the phone is being *held* in an upright angle, the activity is again clearly distinguishable from *lying on desk*. In conclusion it can be said, that the reliability is very individual to each classification.

Therefore, activity recognition is needed, which despite of the noisy sensor data patterns and the differences between the detection of the activities, is **robust**. Otherwise the user would not accept a system, that only under certain conditions can recognize the activities reliable. As sometimes the detection of some activities is very unreliable, there has to be a method to separate the unreliable from the reliable recognitions.

1.2.4 Challenge 4 - Resources

In early days the mobile phone was mainly used only to make calls. Nowadays the devices are also used for various other applications like multimedia, navigation, Internet browsing or taking pictures. Although mobile processors are increasing in speed, their memory gets bigger and even with the new multi core architectures, the user does not want to have the normal usage of the phone limited due to the activity recognition.

More important than the processor load is the battery runtime, since the capacity can not be increased as easily as the processor speed. An increased capacity of the battery can mostly only be reached by increasing the battery size. But with a bigger battery also the weight of the phone increases, which would again lower the user acceptance of the device. The battery runtime is also correlated with the processor load, as more calculation time for the activity recognition also demands more energy.

To reduce the processor load and in conjunction the energy consumption, the activity recognition needs to have as few calculations as possible. If the processor load is kept at a minimum the battery runtime is not significantly lowered due to the activity recognition. Having an activity recognition which is not constantly running could further reduce the energy consumption, since the phone can enter sleep phases where only a minimum of energy is consumed. Reducing the **resource** consumption will increase the user acceptance of the activity recognition.

1.2.5 Challenge 5 - Conditionality

It was already mentioned that the phone can be at various positions on the user or in the environment. Some users carry the phone in the pants pocket, others have it in their jacket or in a backpack. Also the phone can be lying on the desk, table or in the car. All these different locations imply the recognition of similar or very different activities, but the sensor patterns identifying each activity are individual to every position of the phone.

Furthermore, the condition of the user or her environment can be different. E.g. the user is having the phone in her pants pocket whilst *standing*, but the user can stand in a bus, train or building. Each condition implies different sensor patterns, where e.g. standing in a bus results in more noisy ones than standing in a building.

Each of the so called **conditionality** results in individual sensor patterns, on which the activity recognition needs to react onto. Each pattern needs to be represented in the classifier, where the complexity of the recognition nor its accuracy should suffer from an increasing amount of **conditionalities**. Otherwise the solutions for a system coping with many different **conditionalities** would be conflictive with solutions for the other challenges.

1.3 Scope and Methodology

The scope of this thesis is to present a system for activity recognition on mobile phones which can cope with all five challenges. The system needs to be **flexible**, so it can be adapted to changed conditions during runtime. The activity recognition has somehow to be **extensible**, so new activities can be detected if necessary. The recognition has to be **robust**, so even though the sensor patterns are very noisy, the detection of the activities is highly accurate. If in some cases the classification can not be reliable, distinction between reliable and unreliable classifications should be possible to further improve **robustness**. Since the **resources** on mobile phones are limited, the system for activity recognition should have a low processor load and energy consumption. Although there are different **conditionalities** in which the phone can be used or the user can be situated in, the activity recognition still needs to be able to classify the sensor data with high accuracy and low calculation effort.

The proposed system is also evaluated according to these five challenges. Most evaluation is based on data from actual mobile phones - occasionally evaluation results are presented which are based on smart artifacts, which deliver similar accelerometer sensor readings, since these experiments have been conducted before the research focus has shifted to mobile phones, but results are still comparable - and various users activities. To prove the **flexibility** the activity recognition system has to adapt to different sensor patterns, where the approach is compared to a system which was explicitly constructed to classify these different patterns. The **extensibility** is shown on a system which is able to recognize a certain amount of activities and is extended to detect additional activities. To demonstrate the **robustness** of the proposed activity recognition, the system which provides **robustness** is compared to a system without this improvement. The impact of the activity recognition on the **resources** is evaluated according to a *standard* recognition, which is successively transformed into the proposed architecture. Furthermore, the impact due to the activity recognition on the **resource** battery is measured on an actual mobile phone. To analyze the capability of the proposed activity recognition system to deal with a high amount of **conditionalities** without the loss of accuracy or increased complexity, the proposed system is again compared towards a *standard* system.

1.4 Contribution

The main contribution to the field of activity recognition on mobile phones with accelerometer sensors is the modularity of the recognition process. Instead of one monolithic classifier many classification modules are used for activity recognition. This reduces the complexity of the recognition process to conserve **resources** and increases accuracy of the classification. Furthermore, with the proposed modular activity recognition each module can be adapted or exchanged individually to offer **flexibility**. To a modular classifier new modules can be added to detect additional activities and therefor **extend** the activity recognition. Each module can classify activities for a certain **conditionality**, where the complexity is not much increased nor the accuracy significantly lowered.

More contributions to activity recognition on mobile phones are the recurrence and the covariant functions of the classification process. The recurrence stabilizes the classification to offer **robustness** to the activity recognition. The covariant functions are especially suited to classify highly correlated sensor data as provided by multi-axis accelerometers. These also make the activity recognition more **robust**.

The adaption of the classification modules onto changed sensor patterns is needed to offer **flexibility**. Therefore, a novel bit-vector masking is contributed, which delivers the adaption without destroying the original classification capabilities. Also, the masking is not permanent and can be removed at any time to restore the original activity recognition.

The activity recognition is trained according to previously collected and annotated sensor data with a novel machine learning algorithm. This novel algorithm was developed since many improvements to activity recognition exclude other *standard* machine learning. The machine learning needs annotated sensor data from mobile phones. To provide this data and to annotate it several tools were developed. One tool is used to record and eventually annotate sensor data on mobile phones. Another tool is used to visualize the sensor data, transform it to other formats or annotating non annotated data. The actual implementation of the novel machine learning algorithm is then used to train the modular activity recognition. Several implementations of the novel modular activity recognition for various phones provide the capability to evaluate the approach online and to make the recognition available for the common user.

To support the modular activity recognition on the common user's phone, a service was developed, which offers **flexibility**, **extensibility** and **conditionality**. The user just downloads the activity recognition application, collects some sensor data and then the service takes over. Within the service the user's data is combined with data from other users, so the amount of collected data for the individual user is reduced to a minimum. Also, the combination of different users data results in a more variant classifier. The service is always running in the background improving the novel activity recognition for the individual user and the whole community.

1.5 Outline of the Thesis

The remainder of this thesis is structured accordingly:

Second, in this thesis the basic principles of activity recognition and the related work is presented. The section starts with a listing of most of the previous research in activity recognition listed according to the main application fields *elderly care*, *fitness*, *well being*, *health care*, *psychology*, *workflow monitoring*, *memory support* and *groupware*. Furthermore, in this section the various methods of sensor data processing from accelerometer measurement to activity class are listed along systems which implement the respective method. Lastly in this section the various methods for classifier fusion are listed. For each method systems implementing it are briefly described, preferably systems for activity recognition. For all the related work the difference and the similarities to the proposed novel modular activity recognition are described. Also, the listing of methods for sensor data processing and classifier fusion are aligned so the chosen method becomes obvious and reasonable to the reader.

Third, the architecture of the novel modular activity recognition is described in detail. Each step of processing from accelerometer measurement over feature extraction to mapping function and classification is specified. Furthermore, the modularity of the classification is explained and what methods for module scheduling are possible. Since the mapping function is trained with novel machine learning, the algorithm is specified as a whole and all the parts of clustering, linear regression and genetic algorithm are described in detail. At last in this section, all the tools for training, data collection, data annotation, the implementations of the classifiers, the data formats, the training algorithm and the graphical user interfaces (GUI) are described in detail.

Fourth, it is analyzed how the novel modular activity recognition can cope with each of the five challenges **flexibility**, **extensibility**, **robustness**, **resources** and **conditionality** of activity recognition with accelerometer sensors on mobile phones. For each challenge the novel aspect of the modular activity recognition is described and how it provides a solution. Some solutions can interfere with solutions for other challenges, thats why at the end of this section two exemplary tradeoffs are analyzed.

Fifth, the solutions to each of the challenges are experimentally evaluated. For each challenge data sets from actual mobile phones have been collected, which represent some aspect of the challenge. The evaluations are mostly done offline, but in some cases it can be done online. The results of the evaluations are presented, analyzed and discussed in detail. The two exemplary tradeoffs are evaluated at last in this section, where at best a clear value determination for the tradeoff is presented.

Sixth, the service to support the novel modular activity recognition on the common user's phone is explained. The service delivers **flexibility**, **extensibility** and **conditionality** according to the users needs. All components and workflows of the service are described in detail. The main workflow of the service is emulated offline according to sensor data from several users to have a first understanding of the service's capabilities.

This thesis is summarized in the final section. Conclusions made throughout the thesis are further summarized and concluded. Finally, the outlook about the future work indicates at which points the research can and will be taken further.

1.6 Publications

Parts of this thesis have already been published at conferences and workshops, which are listed in the following:

2011

Matthias Budde, M. Berchtold, M. Beigl Video: Activity Recognition on Mobile Phones – Why do we need it and how can it be done?, Video on the 9th International Conference on Pervasive Computing (Pervasive'11), San Francisco, USA, June 12-15, 2011.

M. Berchtold, H. Guenther, Matthias Budde, M. Beigl Scheduling for a Modular Activity Recognition System to Reduce Energy Consumption on SmartPhones, 24th International Conference on Architecture of Computing Systems (ARCS'11), 2nd Workshop on Context-Systems Design, Evaluation and Optimization (CoSDEO), Como, Italy, February 22-25, 2011.

2010

M. Berchtold, M. Budde, D. Gordon, H. Schmidtke, M. Beigl ActiServ: Activity Recognition Service for Mobile Phones, Fourteenth Annual IEEE International Symposium on Wearable Computers (ISWC'10), Seoul, Korea, October 10-13, 2010.

M. Berchtold, M. Budde, H. Schmidtke, M. Beigl An Extensible Modular Recognition Concept that Makes Activity Recognition Practical, 33rd Annual German Conference on Artificial Intelligence (KI'10), Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany, September 21-24, 2010.

M. Berchtold, H. Guenther, M. Beigl A Demonstration of a Robust Context Classification System (CCS) and its Context ToolChain (CTC), Demonstration on the Eighth International Conference on Pervasive Computing (Pervasive'10), 17-20 May 2010, Helsinki, Finland.

H. Guenther, F. El Simrany, M. Berchtold, M. Beigl A Tool Chain for a Lightweight, Robust and Uncertainty-based Context Classification System (CCS), 23rd International Conference on Architecture of Computing Systems (ARCS'10), 1st Workshop on Context-Systems Design, Evaluation and Optimization (CoSDEO), Leibnitz Universitaet Hannover, Hannover, Germany, Februar 22-25, 2010.

2009

M. Berchtold, M. Beigl Increased Robustness in Context Detection and Reasoning using Uncertainty Measures - Concept and Application, Proceedings of the European Conference on Ambient Intelligence (AmI'09), Springer, Salzburg, Austria, 2009.

2008

M. Berchtold, T. Riedel, K. van Laerhoven, C. Decker Gath-Geva Specification and Genetic Generalization of Takagi-Sugeno-Kang Fuzzy Models, IEEE International Conference on Systems, Man and Cybernetics (SMC'08), Suntec Singapore International Convention Exhibition Centre, Singapore, October 12-15, 2008. (slides)

M. Berchtold, T. Riedel, M. Beigl, C. Decker AwarePen - Classification Probability and Fuzziness in a Context Aware Application, The 5th International Conference on Ubiquitous Intelligence and Computing (UIC'08), Oslo University College, Oslo, Norway, June 23-25, 2008.

M. Berchtold, T. Riedel, C. Decker, M. Beigl, C. Bittel Quality of Location: Estimation, System Integration and Application, The 5th International Conference on Networked Sensing Systems (INSS'08) June 17 - 19, 2008, Kanazawa, Japan.

2007

M. Berchtold, C. Decker, M. Beigl, T. Riedel, T. Zimmer Using a Context Quality Measure for Improving Smart Appliances, The 7th International Workshop on Smart Appliances and Wearable Computing (IWSAWC'07) Jun. 29, 2007, Toronto, Canada.

2 Basic Principles and Related Work

The research field of activity recognition considers the recognition of user activities due to sensors attached to or carried on their body. This thesis is in particular about a modular activity recognition especially suited for mobile phones with internal accelerometer sensor. The focus lies on the sensor based single user activity recognition, in particular the accelerometer sensor single user activity recognition is explored. Whereas many sensors do apply for activity recognition, the most common and most researched one is the acceleration sensor. Since this sensor now is standard in most mobile phones, many activities can be recognized with it and its processing can be done resource efficient, the accelerometer is solely used for activity recognition in this thesis. Furthermore, the accelerometer sensor is the most challenging and also the most expressive one for activity recognition. This makes the research, especially of the recognition with mobile phones, very interesting and it is far from complete up to this point.

Activity recognition is a research topic in computer science for many years now. As early as the 1980's this topic is relevant to many disciplines, where early descriptions used the term *plan recognition* [83] or *action recognition* [154] in the fields of knowledge discovery, artificial intelligence, computer vision and robotics. Some of the earliest publications about activity recognition in the field of mobile and ubiquitous computing are [148], [126], [125] and [21]. Where in [148, 21] the location of the user is indicating their activity - although the recognition of activity is not explicitly named, the systems descriptions indicate an activity recognition indirectly -, other sensors besides location sensors are demanded for the recognition of the activity superset context in [126]. The usage of low level sensors for context interaction and activity recognition is proposed in [125], where the authors firstly apply a mobile phone like personal digital assistant (PDA).

First, in this section the various systems for activity recognition are investigated according to their similarity towards the proposed one. Since only a few systems utilize mobile phones as data source or processing platform, a general overview of the systems and approaches concerning activity recognition is presented according to their application field. Many applications in mobile, ubiquitous and pervasive computing utilize activity recognition, where five general application fields (fig. 2) could be identified: (1) **elderly care**, (2) **fitness, well being and health care**, (3) **psychology**, (4) **workflow monitoring** and (5) **memory support and groupware**. Also, many publications do not apply to any field and only focus on finding solutions to the activity recognition challenges. All

Figure 2: Application fields of activity recognition with focus on mobile phones.

known research in mobile, ubiquitous and pervasive computing is listed according to the application field where it is applied. Also, the systems which utilize accelerometer sensors are of most relevance. In this manner a system relying on other sensors is only listed if it is relevant to the respective application field. At the end of this subsection all the relevant and comparable approaches are summarized, where a table gives an overview about the basic concept of evaluation of the respective system and the results.

Second, the general approach and the related work to sensor data processing for activity recognition is investigated. For activity recognition with accelerometer sensors usually one processing stream (fig. 3) is used. At the beginning is the sensor (fig. 3, 1), which is in this case an accelerometer sensor. Then the data windowing and the

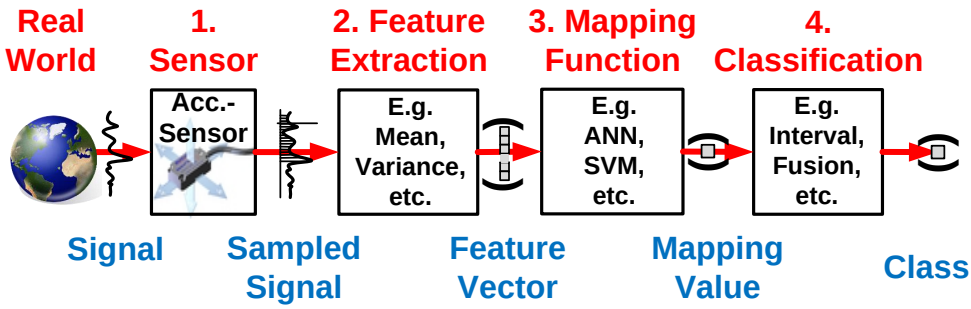


Figure 3: Sensor data processing stream for activity recognition.

feature extraction (fig. 3, 2) reduce the dimensionality and deliver features which are special to the activities which should be recognized. In a next step the mapping function (fig. 3, 3) reduces the dimensionality further to only one, which can then be classified (fig. 3, 4) in the last step. Sometimes the last two steps of mapping and classification are bound together in one algorithm. Algorithms and methods for the main parts of feature extraction and classification are described in the second part of this section. The aim is to identify the best algorithms and methods for the accelerometer data processing, but the focus is clearly lying on the mapping function and classification. The novel modular activity recognition which is proposed in this thesis consists of mapping and classification, where the sensor sampling, the windowing and the feature extraction are firmly predefined so the degrees of freedom of the modular classification can be explored. Along with the methods and algorithms different systems for activity recognition are listed, which implement the respective. Although the main improvement to activity recognition is the modularity, the mapping function and the overall classification process implement other innovations, too. These novelties also help to cope with the challenges of activity recognition on mobile phones. The related work in this subsection is therefore structured so that the shortcomings according to these improvements are obvious.

Lastly, the different methods of classifier fusion are listed and discussed, since the key innovation to the activity recognition in general and on mobile phones in particular is the modularity of the classification process. None of the related methods, which are partly utilized for activity recognition, are matching the proposed modular classification nor can they deliver solutions for all the challenges of activity recognition on mobile phones - at least as known up to this point. Only the proposed modular system is evaluated according to **flexibility**[1], **extensibility**[1], **robustness**[1], **resource**[1] efficiency and **condition**[1] adaptivity.

[1]The bold words correspond to the five challenges of activity recognition on mobile phones: **flexibility**, **extensibility**, **robustness**, **resources**, and **conditionality**

2.1 Activity and Context Recognition

As early as 1999, the project *TEA* [125] proposed methods for recognizing context with low-level sensors, demonstrating the general feasibility on an extended Nokia mobile phone. Over the years this research field increased dramatically and further distinctions of context had to be made. The term activity recognition covers the context information which is more user centered and is mostly recognized due to sensors attached to the user's body. A highly cited publication in the field of activity recognition is [20].

In this thesis the focus lies on activity recognition with mobile phones, since they are always with the user and mostly on the client. Since mobile phones are just arising in the research of activity recognition, most research was done with small sensing devices firmly attached to several body parts.

Another advantage of activity recognition on mobile phones is, that the applications utilizing the activity information can directly be run on the phone. An example can be found in [127], where the ringer volume, vibration or other alerts are controlled by the context information. There are more general application fields for activity recognition not originating or limited, but applicable to mobile phones.

In this subsection all the classical application fields for activity recognition are listed and applications are described which apply to the respective field. This categorization is based on many years of experience in the field of activity recognition, but does not claim to be extensive or complete. Other researchers possibly have other opinions about a categorization, but since no publication could be found which made a definition for the whole field of activity recognition so far, the proposed structure is applied to this subsection. Also, in many application fields only a few publications could be found, which are mostly not very correlated to the approach presented in this thesis, but are listed anyway for completeness. If a system or an aspect of an approach is more related to the system in this thesis, the approach is described in detail according to the similarities. Other approaches which are less correlated are only briefly described, but the aspects which are similar are highlighted.

At the end of this subsection all applications especially related to the approach proposed in this thesis are summarized and a table gives an overview about similarities and differences. Especially the systems utilizing mobile phones as sensor sources or even the activity is recognized on and with the phone are highlighted. Furthermore, the systems which are most related to the one presented in this thesis are repeatedly listed and described again according to the similarities and differences.

Again, the aim in this subsection is to give a general overview of what was done in activity recognition according to the various application fields and research communities. Only a few systems are based on mobile phones and none - at least according to the latest investigation done by the author - are especially suited to cope with the five challenges identified in this thesis. Most of the research done on activity recognition is somehow related, but only a few are directly comparable.

2.1.1 Elderly Care

The application field where the most publications concerning activity recognition are made is elderly care. Since the population of industrial countries is rapidly aging, in several research fields people are researching systems to support the elderly. Activity recognition can be used to monitor home activities, record disease progression or detect falls. In cases of accidents an appropriate alarm can then be sent to call rescue. Dangerous situations could even be anticipated so they can be prevented. Some of the applications of elderly care utilizing activity recognition are listed in [15].

Home Activities There is numerous work on activity recognition for elder care already published. An example is [74], where three triaxial accelerometers were combined with a RFID reader. The accelerometer data was classified on 18 activities with a decision tree and up to 93% accuracy. The high number of 18 activities are only detectable with many accelerometers firmly attached on various body positions in combination with a RFID reader. This scenario is not realizable when only the internal sensors of a mobile phone are utilizable.

An interesting publication which considered both the elderly and the health care scenarios can be found in [72]. In the paper the focus lies on the presentation of a scalable system, where the whole system needs not to be re-modeled or re-trained when sensors change. Also, the gathering of training data is an issue according to the authors. In the paper the issues of activity recognition are discussed more than an actual system presented. The presented approach is based on a multi layer activity recognition system. The issue of scalability is also of concern

in this thesis, but since the authors did not present an actual working system which can be evaluated, the approaches are not comparable.

Disease Progression A system for detecting activities of daily living for elderly or people with chronic diseases is proposed in [11]. The paper focuses on the transitional patterns between activities, so the progression of diseases can be detected. Six different transitions could be detected with an accuracy of 76% based on two different classifiers, a C4.5 decision tree and a KNN classifier. The sensor used for recognition is an e-AR earpeace with an triaxial accelerometer embedded in the device.

Fall Detection Fall detection is an important issue in elderly care, since a lot of severe injuries or even death can result from falls. An extensive survey on fall detection can be found in [6]. The authors first list some definitions of falling and explain these with pictures as well. Also, risk factors for falling are named and causes are listed. The possible causes are grouped in two categories, physical causes and activities. Further sections concern *typical fall scenarios*, *risk assessment tools*, *wireless sensor networks and general system architecture*, *performance evaluation parameters and scenarios*, *fall study database* and finally an *overview of fall detection algorithms*.

An exemplary work on fall detection was published in [100]. The authors used a machine learning approach to recognize seven activities with twelve body tags attached to shoulders, elbows, wrists, hips, knees and ankles. The tag coordinates were acquired through infrared. Since no accelerometers were used, this work is not comparable to the approach presented in this thesis.

In the work of Zhang et al. [165] an one-class SVM algorithm is used to detect falls with one accelerometer sensor. The acceleration sensor was firmly mounted on the users belt buckle. Only two classes were distinguished, a fall or everything else. Data was collected with twelve subjects aged from 10 to 70 years in six different scenarios with different risk factors. Since the one-class algorithm is not usable for detection of more classes and the sensor is firmly mounted on the subjects body, this work's results are only partly comparable to the results presented in this thesis.

A fall detection that is also running on a mobile phone was presented in [164]. Here a combination of an one-class SVM algorithm, a Kernel Fisher Discriminant (KFD) and a k-NN (Nearest Neighbor) classifier was used to distinguish falling from other regular activities. The evaluation includes data from 32 subjects, where 12 subjects had been elder and 20 younger people.

2.1.2 Fitness, Well-being and Health Care

A survey on activity recognition using inertial sensors in health care, well-being and sports was published by Avci et al [15]. The named activity recognition publications are divided in the three groups *medical applications*, *home monitoring and assisted living* and *sports and leisure applications*. A subdivision of the *medical application* into publications dealing with *monitoring and diagnosis*, *rehabilitation*, *correlation of movements and emotions* and *child and elderly care* was made. Also, the category home monitoring and assisted living was subdivided into the categories *traking, monitoring and emergency help*, *assistance for people with cognitive disorders* and *assistance for people with chronic conditions*. The *sports and leisure applications* were subdivided into the categories *daily sports activities*, *martial arts*, *autonomic video annotation* and *performance sports*. A second part of the survey considers the different steps of processing the sensor data and which system implements which method. Finally, a table gives an overview over the listed publications. The structuring of the related work in health care, well-being and fitness does not follow the categorization of the survey paper, but some of the publications listed there are also described in this subsection.

Fitness and Sports In [51] an evaluation of an activity recognition for daily and sports activities is presented. First, the authors applied various feature extraction methods of which they selected seven which are of highest relevance to detect the activities. The features were mostly calculated over accelerometer signals provided by two sensors applied to the users hip and wrist, but also the ground speed was provided as feature by a GPS device. The data was stored on the GPS device and evaluated offline on a server. For classification four different methods were used: a custom decision tree, an automatic decision tree, an artificial neural network and a hybrid model. The approach was evaluated with 12 different users and 68 hours of data, of which 21 hours had been from a controlled

and 47 hours of data from an uncontrolled environment. A recognition rate of 89% (75%) was reached with a training data set mixture of supervised and unsupervised conditions.

In [99] a Philips NWS Activity Monitor was used to detect five daily and sport activities. Nineteen different features were used out of the categories standard deviation, frequency-domain entropy and orientation variation. Also a PCA was used to filter out the correlated features and to reduce the amount of dimensions. For classification a Bayesian decision process was used, which was evaluated with two different methods: the ”leave-one-out” cross validation and the ”10-fold” cross validation. For 24 users and five activity classes a recognition rate of nearly 80% could be reached. The results were compared to a decision tree approach, where the Bayes method showed similar performance, but offered a more extensible algorithm structure. The system was used to calculate the test persons calorie expenditure per day.

A system to detect Wing Tsun (martial arts) movements is presented in [71]. Since the authors not only used eight 3-axis accelerometers, but also 3-axis gyroscopes and magnetometers, this work is hardly comparable to the mobile phone systems applied in this thesis. The eight sensor boxes were carried on the wrists, the lower legs, knees, the neck and the rear hip. A system to annotate sports videos is presented in [39]. Since the presented system was designed to recognize gestures, but this thesis focuses on activity recognition, the results presented are not comparable.

30 gymnasium activities were recognized with five firmly attached acceleration sensors and a heart rate monitor in the work of Tapia et al. [137]. For training the classifiers and evaluate the approach a data set from 21 subjects performing the 30 activities had been collected. The system performs very well, but due to the high amount of used external acceleration sensors, which were firmly attached to the different body positions, the approach is hardly comparable. Also, the authors used an external heart rate monitor as well, which further improved the recognition results.

Nine everyday and fitness activities were detected through the activity recognition system presented in [118]. The system uses six Inertial Measurement Units (IMU) which include triaxial accelerometers, gyroscopes and magnetic field sensors. Due to the high amount of IMUs used, which were firmly attached to five upper body positions and a foot, the recognition rates were, despite the high amount of recognized activities, very good. This accuracy is hard to reach with an activity recognition which only uses the internal acceleration sensor built in a mobile phone, but nonetheless the novel architecture of the system presented in this thesis can reach this upper limit under certain circumstances.

Well-being The correlation of well-being and physical activities is highlighted in the work of Meyer et al. [105]. The authors claim, that the users well-being could be improved, when medical and activity data is shared in blogs or fora, since early symptoms of diseases can be captured through the community. How the activity information should be gathered is not explained in the paper.

Health Care An activity recognition system based on accelerometer sensors and RFID readers to calculate expended calories for a health care scenario was proposed by Hong et al [75]. User activities of daily living were detected via two accelerometers, firmly positioned at the waist and above the knee. The recognition rates for 15 subjects and 5 classes for the acceleration sensors were 95% accurate using a decision tree classifier.

A smart buckle was used in [43] for activity recognition in medical monitoring applications based on a 3-axis accelerometer sensor in combination with an optical sensor in a fixed position. Conclusive information about the overall recognition rates could not be found in the paper.

A triaxial accelerometer located firmly on the dominant thigh was used in combination with a heart rate sensor in [101]. For the six test subjects an average recognition rate of 85% for eight classes was achieved.

The work of Roggen et al. [120] is aligned on two application fields, industrial wearable assistance and pervasive health. Two issues are addressed, sensor configurations and variability in labels for activity recognition. The authors claim, that sensors such as accelerometers can not be placed without variance in a daily life setting. Some of the same authors published a method which deals with this issue in [161]. Other work dealing with variant phone orientations and placements can be found in [91] and [92]. The issue of labeling concerns the annotation of sensor data in complex multi sensor scenarios, which is addressed in [142]. Both issues are of concern in this thesis, too.

A system for activity recognition in three health and fitness related usage scenarios is presented in [94]. Mainly the sensor platform is presented with some initial results of activity recognition. A more detailed description of the

algorithm and evaluation used can be found in [95].

In [35] activity recognition on mobile phones for health care is introduced. The system is called DiaTrace, which was run on a mobile phone and only utilized the internal accelerometer sensor. The application included an activity counter which reminds the user to do additional activities, the activities can be shared with friends over social networks and the daily calorie consumption can be calculated. Any details about the activity recognition algorithms and the feature extraction are not made. Also, only a rough number about the overall recognition accuracy could be found. These limitations in description detail make this work hardly comparable to the approach proposed in this thesis.

Another system utilizing accelerometer sensors in a health care setting is proposed in [128]. An accuracy of 96% was achieved for 6 classes with a sensor fastened to the user's wrist. As most activity recognition publications in health care also in [128] accelerometer sensors on fixed body positions were used. Only one activity recognition system for health care utilized mobile phones and their internal sensors [35].

2.1.3 Psychology

The usage of computers and mobile devices in psychotherapy and psychology is becoming a valid research area. Further, mobile devices to support the patients have been looked at in the work of Sá et al. [50]. The authors claim that handheld-based tools let therapists customize psychotherapy for individual patients. The paper focuses on tools to customize questionnaires for mobile devices, so the patient can immediately react, when symptoms occur. No actual sensor-based recognition was performed or mentioned throughout the paper, but it emphasizes the usage of mobile devices in psychotherapy.

Early Diagnoses of Psychiatric Diseases A system to infer early diagnosis of psychiatric diseases through activity and emotion recognition is presented in [134]. The authors claim, that the sensing system needs to be as unobtrusive as possible, because otherwise the patients will reject it. Also, for the same reasons the patients can not be asked to perform any kind of training. In this manner a watch like sensing platform was used, which previously had been presented in [119]. The software architecture is called *Psychiatric Patient centric Pervasive (P-cube)* platform and was installed on the patients personal device, e.g. their smart phone or laptop. The activity recognition was done according to accelerometer, microphones and location measurements which were provided through wrist and hip worn sensor devices. No accuracy percentage of the activity recognition or of the overall system are presented in the paper, so the approach can not be compared to the one presented in this thesis. The authors also presented a system for early recognition of transitions between normal, manic and depressed phases to detect a patients bipolar disorder [135]. Also, no comparable numbers are shown in this paper

Monitoring A system to monitor children with autism is shown in [151]. The system was mainly used to detect self-stimulatory behaviors with three triaxial accelerometers attached to the patients right wrist, back and left ankle. Seven self stimulatory activities such as *drumming*, *pacing* or *rocking* were detected and separated from any other normal activity combined in a *null class*. The self stimulatory (short: stimming) activities were performed by a researcher inbetween normal activities like *walking* or *writing*. In 15 trails a set of 105 stimming examples had been collected. The data was transmitted over Bluetooth, centrally collected on a laptop computer and processed offline using a HMM classifier. Even though the authors presented a method of detecting stimming activities with an accelerometers, the recognition results ($> 90\%$) are not directly comparable, since three firmly attached sensors were used and the HMM classifier method classifies on time series of sensor data. In this thesis a classifier approach is used, which also can classify activities with nearly no movement, where only the orientation of the sensing device is indicating the activity. Further, only data of one test person was used in their work to evaluate the approach.

Emotion Recognition Mobile phones were used in the work of Rachuri et al. [115] to conduct social and physiological experiments in an unobtrusive way. The system presented was used to detect users emotions and activities with several kinds of built in sensors, such as microphone, Bluetooth user proximity, GPS and accelerometers. Since the primary focus of the work was not lying on the activity recognition with accelerometers, only two classes were distinguished, movement and non-movement. Particular results on the performance of the detection of these two

activities were not presented, but the authors claim that the stronger CenceMe [106] (listed under *Miscellaneous: Frame Works 2.1.6)* application could be included in the EmotionSense application.

A hierarchical classifier approach to detect emotions is presented in [10]. Since in their approach voice-based human-machine interaction as audio data source was used, the work is not directly comparable to this accelerometer based activity recognition approach.

Another approach to detect human emotions is presented in [124]. The authors used an biologically inspired neural model to detect the emotions according to pictures of the humans body pose. Since the approach was using images as data source, the research is not comparable to the one presented in this thesis.

2.1.4 Workflow Monitoring

Another very relevant application field for activity recognition is workflow monitoring. Various different activities can be monitored in many different jobs to e.g. detect faults in behavior of mechanics so they can immediately be corrected, detect or predict accidents so help can be called automatically or accidents even be prevented. Important work in this application field was done in the *WearIT@Work* project [4].

Industrial Environment An automatic workflow monitoring in industrial environments based on visual sensors is proposed in [145]. The proposed approach consisted of an robust scene descriptor and an efficient time series analysis method. The visual sensor is different in processing a classification than the used accelerometer in this thesis, so the approaches are not comparable.

An approach which also uses modularity to detect 20 activity classes in a car assembly workflow is described in [110]. The authors utilized a jacket with seven IMUs (accelerometers and gyroscopes combined), two sleeves with eight channel FSRs (force sensitive resistor) and an UWB (ultra-wide band) relative positioning system. Eight test subjects were used to simulate the car assembly tasks previously observed from a worker at a Škoda production facility. The proposed approach is modular in a way where it separates the event streams, so many independent spotting processes can be used. The authors claim, that this approach allows the easy addition or removal of classes, which would be similar to the proposed approach in this thesis. The biggest differences are on the one hand the classifiers that had been used, which were combinations of string matching, k-NN and Bayes classification, and on the other hand the used sensor jacket, which utilized a huge amount of sensors which are firmly attached to the different body parts. The modularity used by the authors is very heterogeneous and does not deliver a reproducible approach for other settings. The classifier they used was tailored to their sensing system and would need to be redesigned for every new one. This is different with the modular activity recognition system proposed in this thesis, which can be considered as a black box and is entirely specified through machine learning. Also, the high amount of firmly attached and various kinds of sensors on different body parts are just not realizable with one mobile device.

Manual tasks of maintenance and assembly were the focus in the work of Ward et al. [149]. In particular the authors tried to detect activities characterized by hand motion and respective sounds. Two accelerometer sensor devices were mounted at two positions of the user's arm. Also microphones were used to detect the accompanied sound. The features peaks, mean amplitude of the peaks and raw sensor values from the x-axis were used as input of a HMM classifier. In total the authors tried to recognize nine activities like *hammering*, *sawing*, etc. in a mockup wood workshop assembly scenario. The two sensor types were classified separately and the results than fused according to highest rank, borda count or logistic regression. Although, the accelerometer classifier was analyzed separately and in conjunction with the audio classifier, clear numbers of the performance of each part are hard to extract from the paper. Further, the firm positioning of the external sensor devices on the users arm and the combination with microphones makes this approach quiet different from the one proposed in this thesis.

Medical Environment A workflow monitoring based on 3D motion features provided by a multiple-camera system is described in [112]. A workflow-HMM was used to recognize activities in a mock-up operating room. Since the used sensors are visual and not motion based accelerometers, the publications are not comparable to the approach proposed in this thesis.

In [122] a Hidden Markov Model (HMM) is proposed for activity recognition in Ambient Intelligence Environments. The trial included the monitoring of five nurses, five medical interns and five physicians through observants in a medical environment, which resulted in 196 hours of data. The observants manually recorded actions, used artifacts, content of conversation, etc., which were used as input for the HMM to estimate six activities. Since no

actual sensors were utilized, the approach is also not comparable to the one proposed in this thesis, but the study could indicate which sensors would be useful to actually detect the workflow in a medical environment.

The workflow of medical personal was also analyzed Vankipuram et al. [143]. The authors used RFID tags and observation for automated workflow analysis and visualization in clinical environments. Since this is more of a positional paper, where the evaluation is only partially based on actual sensors, which are also different from the one used in this thesis, there is no common base to compare these approaches.

2.1.5 Groupware and Memory Support

The last application fields for activity recognition presented here are groupware and memory support. In these fields nearly no applications utilizing activity information exist so far. Since these will be relevant application fields for activity recognition in future, especially with mobile phones (extension for applications like calenders or social networks), this field should become of more interest to the community.

Groupware In [40] a system is presented where different context sources were combined to create story board like views of recent activities. Generally two information sources were distinguished, white and black box. The white box sources can directly be used for reasoning, where black box content can only be interpreted by humans. As a source of white box content groupware is listed. Black box sources were video and audio streams. Since no accelerometer based activity recognition was presented, this work can not be compared to the approach proposed in this thesis.

Memory Support Memory support was the application for activity recognition shown in [88]. The authors tried to annotate videos with activity data to create daily life stories to support dementia patients. The activity information had been gathered through a logging application, which recorded GPS locations, Bluetooth MAC addresses and video data. The activity was then inferred from the location of the user and the video content. No actual evaluation of the approach is presented in the paper. Since the authors did not use an accelerometer sensor for activity recognition this work is not comparable.

Originally positioned in the application field of groupware the system presented in [40] could also be categorized as memory supportive. An early appliance utilizing not activity but context information was the MemoClip [21]. The appliance reminded the user at certain locations such as toilet that the user had to wash hands. The relevant places were tagged with a sensor device and the remembrance text was displayed via a badge attached to the users chest pocket. None of these systems utilize accelerometers and thus are not comparable.

2.1.6 Miscellaneous or not bound to any Application Field

Some of the publications in the field of activity recognition are not particularly bound to any application. Especially the research done on activity recognition in artificial intelligence investigates the capabilities of the proposed approach without naming any application. Also, there are a lot of frameworks which could be suited for nearly any application, but merely investigate the support due to activity recognition. In the frameworks the activity recognition is mostly only a part of the derivation of several other context information such as location or proximity.

Pervasive, Mobile and Ubiquitous Computing [20] is a highly cited publication of activity recognition in pervasive computing using acceleration sensors located on four limb positions and the hip. Four classification algorithms were compared, where the best performing algorithm (C4.5) had an average recognition accuracy of 73% for 20 activity classes. Because their approach used 5 acceleration sensors in fixed positions with classification done at the back-end, the results are not directly comparable to the one presented in this thesis.

In [103] the *eWatch* sensing platform was used to detect six activities. The *eWatch* was carried in the test subjects pocket among other body positions. For six users a recognition accuracy of 80% was reached. Although the classification was done on the *eWatch* and the position is not fixed, the amount of recognized classes can be exceeded in this thesis by up to ten and higher recognition accuracies can be reached.

A mixture of dynamic and static activities were recognized according to accelerometer sensor data provided by an Android based cell phone in [93]. The six activities were recognized with three different algorithms: decision trees (J48), logistic regression and a multilayer neural network. In the evaluation with sensor data from 29 users

the multilayer neural network was the most accurate of the tested classification algorithms. Since the recognition is based on accelerometer sensor data from a mobile phone, this work is one of the most related. Nevertheless, in this thesis a novel modular activity recognition is proposed, which is evaluated according to five challenges, where the approach presented in [93] is only evaluated according to accuracy of the recognition process.

[36] and [66] both present activity recognition systems for mobile phones, where Brezmes et al. [36] only used sensors and resources native to the phone for sensory acquisition and classification, Györbíró et al. [66] also utilized a wristband as external sensor. In [36] just six activities were distinguished with only 80% recognition accuracy in average. In [66, 52] a wristband in combination with a mobile phone was used also to recognize only six activity classes. Both approaches do not qualify for modular classification, were not evaluated according to the five challenges and they do not recognize a significant amount of activities with high accuracy.

Artificial Intelligence Generally all activity recognition algorithms belong to the field of pattern recognition, which is part of artificial intelligence (AI). Here publications are cited, which were published not in mobile, ubiquitous or pervasive computing, but directly on AI conferences. These papers mostly contain work of a more algorithmic nature, where the evaluation was mainly used to validate the approach and not to present a system for everyday use.

An example of research in the field of artificial intelligence is [117], in which a triaxial accelerometer was used to detect eight activity classes. Several algorithms in various combinations and versions were investigated in four different settings on their classification accuracy. Since the employed sensor is mounted on a fixed position on the test persons pelvic region and the algorithms that were used do not qualify for modular and flexible usage, the recognition rates are not directly comparable. Nevertheless, with the approach presented in this thesis, their recognition rates (91% for eight activity classes [117, *setting 3*]) can be outperformed.

Activity recognition based on fuzzy classifiers is presented in [73] and [155]. Helmi et al. [73] used a Fuzzy Inference System (FIS) and Yang et al. [155] utilized neuro-fuzzy classifiers. Helmi et al. reached a recognition rate of 95% for four classes and three subjects, where Yang et al. eight classes for seven subjects were able to detect with 93% accuracy. Both used external sensors on fixed body positions and the recognition was not done on embedded or mobile devices. Since in this thesis a FIS is also used in the recognition process, the work of Helmi et al. [73] is somewhat relevant to this approach, but no modularity in the recognition was used and the sensors were firmly attached external ones.

Frame Works *BeTelGeuse* [89] is a data collection framework for mobile devices that automatically infers context from sensor data. It supports receiving data from internal phone sensors, external Bluetooth-enabled sensors or data sources from the web. The design focus was to create a freely available, extensible and platform independent system. Context recognition is encapsulated in separate plug-ins, which are able to recognize basic activities from accelerometer and/or heart monitor data. No evaluation solely for the activity recognition was presented in the publications, nor any novel algorithm for accelerometer data classification on mobile devices.

Another sophisticated personal mobile sensing system with only limited impact on the phone's functionality is *UbiFit* [45]. The *CenceMe* [106] project uses off-the-shelf mobile phones to infer a person's context and share the collected information with a social networking community. *CencMe* had been also one of the first people-centric applications that had been validated through an extensive user study. Wearable sensors – those of the *Nike+iPod Sport Kit* – are also employed in the *iLearn* system [123] for Apple's *iPhone*.

The context-aware mobile phone *SenSay* [127] was designed to, among other features, automatically turn the ringer on and off to prevent inaudible ringer volume in a loud environment. However it requires an external sensor box on the user's hip as well as ambient microphones on their bodies. Another approach which uses special wearable sensors is the *Mobile Sensing Platform* (MSP) [44], which was specifically developed for activity recognition. It had been refined over a period of four years and deployed in several applications, such as on-body sensing, real-time activity inference in body-workout situations or an application that supported people in recalling what physical activities they had been engaged in during the last week. Because of its long duration, the depth of the project is very high. The involved researchers reported on many lessons they learned in the course of their work. But, no novel algorithm for activity recognition was presented.

In this thesis, the focus lies on the improvement of the recognition process to cope with the five challenges, which have been identified before, of activity recognition on mobile phones. With the frame works design, which

were presented so far, were focused more on platform independence and general applicability than on the improvement of activity recognition on mobile devices.

2.1.7 Summary and Overview

Table 1 contains an overview of related work published on the topic of activity recognition utilizing accelerometer senors. Publications are grouped according to the research field in which the work was conducted. The table lists

	# recog. classes	accuracy(%) - claimed	accuracy(%) - recalc.	# test subjects	# acc. sensors	other? yes(y)/no(n)	internal(i)/external(e)	fixed(f)/non-fixed(n)	server(s)/device(d)
Elderly Care									
Hong et al. [74]	18	93	92	15	3	y	e	f	s
Ali et al. [11]	6	90	75	5	1	n	i	f	s
Zhang et al. [165]	2	97	94	12	1	n	e	f	s
Zhang et al. [164]	2	85	-	32	1	n	i	n	s
Fitness, Wellbeing and Health Care									
Ermes et al. [51]	9	89	75	12	2	y	e	f	s
Long et al. [99]	5	80	78	24	1	n	i	f	s
Hong et al. [75]	5	94	-	15	2	-	e	f	s
Cho et al. [43]	9	-	-	-	1	-	e	f	s
Maguire et al.[101]	8	85	-	6	1	-	e	f	s
Song et al. [128]	6(9)	96	-	-	1	-	e	f	s
Tapia et al. [137]	30	98	-	21	5	y	e	f	s
Reiss et al. [118]	9	97	-	-	6	y	e	f	s
Lester et al. [94][95]	10	95	-	2	1	y	e	f	s
Bieber et al. [35]	6	>95	-	-	1	n	i	n	d
Psychology									
Westeyn et al. [151]	8	91	-	1	3	n	e	f	s
Workflow Monitoring									
Orgis et al. [110]	20	-	-	8	7	y	e	f	s
Ward et al. [149]	9	98 (87)	-	5	2	y	e	f	s
Misc., Interdisciplinary or Independent									
Bao et al. [20]	20	84	73	20	5	n	e	f	s
Maurer et al. [103]	6	80	-	6	1	y	i	n	d
Helmi et al. [73]	4	95	-	3	1	n	e	f	s
Ravi et al. [117]	8	91	73	2	1	n	e	f	s
Kwapisz et al. [93]	6	92	80	29	1	n	i	n	s
Brezmes et al. [36]	6	80	-	-	1	n	i	n	d
Györbíró et al. [66][52]	6	82	-	3	3	n	e	f	d
Saponas et al. [123]	4	97	-	8	2	n	i/e	f/n	d
Siewiorek et al. [127]	4	-	-	-	2	y	e	f	s
Consolvo et al. [45, 44]	5	94	77	12	1	n	e	f	d
Sun et al. [133]	7	95	-	7	1	n	i	n	d

Table 1: Overview: publications about activity recognition utilizing accelerometer sensors.

the relevant underlying conditions, such as if the sensor was internal or external (i/e), additional non acceleration sensors (yes (y) / no (n)) were used, sensor(s) were fixed to a position (f/n) or if the sensor data was processed on the device or a server (d/s). The most related publications, which were using a mobile phone as sensor source or for the entire activity recognition, are marked red. Also, the overall recognition accuracy as it was specified in the respective publication and how it is with the metric used in this thesis are listed in the table. Furthermore the number of recognized classes, test subjects and accelerometer sensors are quoted.

The novel modular activity recognition proposed in this thesis is running solely on the mobile phone, using only the internal accelerometers provided. It does not rely on fixed sensors, can recognize a variable number of dynamic and static activities (tests with 9-15) and is partly evaluated with a fairly high number of test subjects (up to 20). Only a few other systems were based and evaluated on the same conditions.

The system proposed in [103] also used a device which was not fixed to any position, utilized the internal sensors and calculated the recognition on the device. However, the system used also another sensor (light), the used device

was not a mobile phone and the amount of activity classes the system was evaluated on is lower than with any evaluation in this thesis. Activity recognition solely using a mobile phone and the internal acceleration sensor was proposed in [36], but no specification about the amount of test subjects was made nor is the evaluation very detailed. A mobile phone was used for activity recognition in [66], but the system utilized three firmly attached external accelerometer sensor devices. In [123] a mobile phone was also used for calculating the activity recognition, but in addition to the phones accelerometer an external one in the users shoe was used.

In [127] the profile of a mobile phone was adjusted among others through activity recognition, but the system utilized external firmly attached accelerometer senors. Also, the system is mainly relying on the context recognition due to audio sources, thats why no numbers about accuracy of the accelerometer based activity recognition were published. A mobile phone in combination with an external firmly attached accelerometer sensor device for activity recognition was also used in [45, 44].

The approach proposed in [93] was evaluated on data from a mobile phone, but nothing indicates the actual implementation of the recognition on the device and the amount of recognized classes is too low for comparison. Also, in [35] only the internal accelerometer sensor of a mobile phone was used for the activity recognition on the device. There is no description of the activity recognition algorithm nor are there detailed evaluation results. In [93] and [35] the underlying conditions are the same as in this thesis.

The work that comes closest to the one presented in this thesis is presented in [133]. Their system also ran on mobile phones and only used the internal accelerometer for the activity recognition. Even though their system reached a **robust** activity recognition with high accuracy and considerd different **conditionalities** of the phone and the user, the system was not evaluated on the other challenges. Further, the approach presented in this thesis was mainly published earlier.

Although the numbers in table 1 could indicate that some other approaches come close to the proposed modular activity recognition, the real strength lies in coping with the previously defined challenges of a recognition on mobile phones. As known so far - this the opinion of the author according to an extensive literature investigation - none of the other systems are as **flexible**, **extensible**, **robust**, **resource** gentle and **condition** supportive than the modular activity recognition proposed in this thesis. Some could be extended by some of the improvements presented in this thesis, but state of the art is, that the investigated approaches currently are not able to deliver what the novel modular activity recognition can. Especially since no other existing system can deliver solutions for all challenges together.

2.2 Sensor Data Processing

The sensor data processing is the key mechanism of any activity recognition. It starts with sampling the sensor, then a windowing of the measurements, furthers a feature extraction delivers classifiable vectors and finally the classification takes place. All steps of processing are graphically described in figure 3. The classification itself is sometimes partitioned into mapping function and actual classification, which is the case in this thesis. The mapping function is used to project the feature vector onto a classifiable one dimensional set and the classification does the rest.

In this subsection the related work to the components of the sensor data processing queue are presented along with the listing of system implementations which utilize them. The first two steps of the processing are not part of this subsection, since the accelerometer has already been determined and the various windowing methods are a complex topic which would unnecessarily increase the degree of freedom to be investigated. Also, with most modern mobile phones the sampling rate of the accelerometer is not adjustable anyway, which reduces the possibilities of the windowing. The list of feature extraction and classification methods covers the most relevant algorithms, but is not extensive nor complete. The structure is aligned so that the methods of choice are obvious to the reader.

This subsection starts with the various methods used for feature extraction, from which two are picked which seem to be the most relevant ones. A closer look is taken into the classification methods and mapping functions, since they are this thesis main focus after the modularity of the recognition process. At the end of each part the key aspects are summarized and the methods of selection are argued and discussed with respect to the challenges of activity recognition on mobile phones.

2.2.1 Feature Extractions

The feature extraction is the first step of algorithmic processing of the accelerometer data in activity recognition. This extraction delivers the relevant features for the classification. The feature extraction is used to reduce the dimensionality of the sensor data stream. Feature extraction methods are listed according to their relevance to activity recognition in *health care*, *well-being* and *sports* applications in [15]. A large set of feature extraction and preprocessing techniques are also analyzed and experimentally evaluated in [55].

Mean One of the standard methods of feature extraction is the mean value. With this value the orientation and the position of the sensing device can be determined with acceleration sensors. The statistical mean is calculated as follows:

$$m(\vec{\mathbf{x}}) = \frac{1}{n}\sum_{i=1}^{n} x_i, \text{ with } \vec{\mathbf{x}} = (x_1, x_2, .., x_n)$$

The mean value has the advantage, that the calculation effort to derive it is minimal. Only the sum of all measurements in one window and one multiplication needs to be calculated.

This feature extraction method was used in many publications about activity recognition, since it is so simple and expressive. In [20] the usefulness of this feature among others was demonstrated. The authors used different classification methods, such as Decision Table, IBL, C4.5 and Naive Bayes to classify the feature vectors. In [56] a genetic programming (GP) was used to determine a function that maps the accelerometer sensor signal onto classifiable feature vectors. The GP selected the best combination from a set of operations and features in order to reduce calculation effort and to gain a recognition robust to sensor displacement. The mean feature in combination with others had the best performance. Also, [101], [117] and [113] make use of this feature extraction method. Since this is a very popular feature for activity recognition with accelerometer sensors, there would be many more to be named, but since the feature extraction is not in the focus of this thesis, the named should be enough.

Root Mean Square Also frequently used in activity recognition is the feature root mean square, which is calculated accordingly:

$$R(\vec{\mathbf{x}}) = \sqrt{\frac{\sum_{i=1}^{n} x_i^2}{n}}, \text{ with } \vec{\mathbf{x}} = (x_1, x_2, .., x_n)$$

This feature again provides no additional or entirely new information to the activity recognition process. If the mean is already a part of the feature vector the extension of the vector by the root mean square would not particularly

benefit the activity recognition, since it is mathematically close to it. The only difference is, that the root mean square is always positive, where the mean can be both. This is also the reason, why the mean in this thesis is chosen over the root mean square, since with the mean feature activities which differ due to the orientation of the device can be distinguished. On the other hand, when such activities would not need to be distinguished, the root mean square would provide a feature that is partly orientation independent.

Publications in which feature vectors with the root mean square were used are [156], [103], [147] and [117], which is surprising, since the combination with the mean is redundant. Also, in [60] the root mean square was combined with the mean feature for activity recognition. In [58] the feature was used to detect wrist rotations of golf swings.

Signal Energy A feature that can bee seen throughout the literature is the signal energy. The signal energy can be used to distinguish sedentary activities from vigorous activities [156][15]. The signal energy is calculated by deriving the integral over the sensor signal, which with discrete values results in the following equation:

$$E(\vec{\mathbf{x}}) = \sum_{i=1}^{n} x_i^2 = \|\vec{\mathbf{x}}\|^2, \text{ with } \vec{\mathbf{x}} = (x_1, x_2, .., x_n)$$

Due due the squared elements of the input vector, the negative part of the signal is practically mirrored along the x-axis. Therefore, in difference to the mean feature the signal energy is always positive. Since the signal energy does not particularly indicate a special movement nor can it be used to determine the device orientation, the usefulness of this feature is questionable. the calculation effort for deriving this feature is not very high and lies somewhere in between the mean and variance calculation.

In [80] the signal energy was analyzed among other features on the usability for activity recognition. Further, in [98] the signal energy was combined with the mean and variance in a feature vector for on-body activity recognition. The feature combination and selection with a genetic programming (GA) was analyzed in [56], where the signal energy was not among the best performing feature combinations.

Signal Magnitude Area A feature that is closely related to the signal energy is the signal magnitude area. Practically, instead of only calculating the area of one accelerometer axis integral, the integrals for each axis are summed up and divided through the number of samples. The signal magnitude area is calculated accordingly:

$$S(\vec{\mathbf{x}}, \vec{\mathbf{y}}, \vec{\mathbf{z}}) = \frac{1}{n} \left(\sum_{i=1}^{n} |x_i| + \sum_{i=1}^{n} |y_i| + \sum_{i=1}^{n} |z_i| \right)$$

Compared to the mean, the signal magnitude are is always positive. Also, since all axes are included in the derivation of one feature, the orientation of the sensing device can not be derived from this feature. The distinction of activities with different device orientations is therefor not possible. In this thesis this limitation for the activity recognition is not tolerable. A combination with a feature which includes this information, the mean feature, would result in redundancy due to the similarity of the features. The information a signal magnitude area can give, can also be derived through the actual classification process.

The signal magnitude area was used as main feature in [156]. In [82] this feature was used for ambulatory monitoring of patients.

Variance Another standard feature used for activity recognition is the variance of the accelerometer data. The variance is of great relevance for detecting non static activities. Especially when the variance is combined with the mean it needs low computational resources. The variance feature is calculated accordingly:

$$\sigma_n^2(\vec{\mathbf{x}}) = \frac{1}{n} \sum_{i=1}^{n} (x_i - m(\vec{\mathbf{x}}))^2, \text{ with } \vec{\mathbf{x}} = (x_1, x_2, .., x_n)$$

Another method of deriving the variance of a signal which is used throughout the literature is calculated as follows:

$$\sigma_{n-1}^2(\vec{\mathbf{x}}) = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - m(\vec{\mathbf{x}}))^2, \text{ with } \vec{\mathbf{x}} = (x_1, x_2, .., x_n)$$

The first method is the biased and the second the unbiased variant. For the feature extraction of activity recognition the biased method is the more relevant. Both methods are very efficient when combined with the mean feature, since the calculation of the variance includes the mean.

A example of a activity recognition system that uses the variance feature is given in [108]. There the variance is used among other features for distributed activity recognition. Another publication where the usage of the variance feature is analyzed can be found in [14]. There its used to evaluate the variable accelerometer sensor placement again for activity recognition. In [79] the features mean and variance were used to detect daily activity patterns with topic models.

Standard Deviation The standard deviation is used much more often throughout the literature than the variance, but is basically the same feature. The only difference is that the standard deviation uses the square root of the variance as shown in the following:

$$\sigma_n^2(\vec{\mathbf{x}}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - m(\vec{\mathbf{x}}))^2}, \text{ with } \vec{\mathbf{x}} = (x_1, x_2, .., x_n)$$

Since there are two methods for a variance to be calculated, both biased and unbiased, there is a second one for the standard deviation, too. The unbiased standard deviation is calculated accordingly:

$$\sigma_{n-1}(\vec{\mathbf{x}}) = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - m(\vec{\mathbf{x}}))^2}, \text{ with } \vec{\mathbf{x}} = (x_1, x_2, .., x_n)$$

Compared to the variance, the standard deviation is a little more calculation intensive since it involves the square root. For the activity recognition it is generally irrelevant if the one or the other is taken, but statistically there is a difference. The standard deviation is a bit more intuitive for describing probability distributions. Also, the standard deviation is in the same units as the measurements of the acceleration sensor which are the input of the feature extraction. Therefore, if the feature extraction should be visualized or directly be interpreted, the standard deviation should be chosen over the variance. In the case of this thesis, the variance is chosen over the standard deviation, since the calculation effort is less and **resources** are limited on mobile phones.

Different feature classification algorithms for activity recognition are presented in [101]. The standard deviation is used among other features. A feature learning algorithm is shown in [114], where one of the features was the standard deviation. An approach where a discrete cosine transformation (DCT) and support vector machines (SVM) were compared towards *traditional* methods like the standard deviation for the purpose of activity recognition was published by Ploetz et al. [114]. Other publications which use the standard deviation as feature extraction in activity recognition are [117] and [113].

Mean Absolute Deviation (MAD) The mean absolute deviation is another feature which is used for activity recognition. This feature is the mean of the absolute deviation, which is calculated accordingly:

$$D(\vec{\mathbf{x}}) = \frac{1}{n}\sum_{i=1}^{n}|x_i - m(\vec{\mathbf{x}})|, \text{ with } \vec{\mathbf{x}} = (x_1, x_2, .., x_n)$$

As can be seen, the mean absolute deviation has not much difference to the standard deviation and the variance. Instead of multiplying each distance of vector element and mean with itself, the absolute value is derived. Therefore, the mean standard deviation needs less computational resources than the features variance and standard deviation. The combination of variance, standard deviation and mean absolute deviation would therefore make no sense and the choice should be made for either one of these features. Further, the combination with the mean feature in a feature vector would enable the mean absolute deviation calculation to reuse the calculated mean.

The mean absolute deviation was used by Yang et al. [156] for activity recognition. The focus of this publication was on effective learning algorithms for constructing neural classifiers. In [103], [147] and [117] the mean absolute deviation was used as feature extraction method. In this thesis the variance feature was chosen over the mean absolute deviation, since its more broadly used and researched. However, the exchange of these two features would result in equal recognition results.

Signal Correlation Another feature, that is frequently used, is the signal correlation between different axes of the used acceleration sensor. The output of such a feature calculation is, how correlated the two input signals are. The feature ranges from total correlation to absolute anti-correlation of the signals. The correlation is calculated accordingly:

$$C(\vec{\mathbf{x}}, \vec{\mathbf{y}}) = \frac{\langle (\vec{\mathbf{x}} - m(\vec{\mathbf{x}})), (\vec{\mathbf{y}} - m(\vec{\mathbf{y}})) \rangle}{\|(\vec{\mathbf{x}} - m(\vec{\mathbf{x}}))\| \, \|(\vec{\mathbf{y}} - m(\vec{\mathbf{y}}))\|}$$

Compared to the previously discussed features, the signal correlation is more calculative-intensive, but again if it is combined with the mean extraction the results can be reused. For this thesis the explicit derivation of the signal correlation is covered due to the implicit covariant membership functions of the classification. Therefore, this feature is included in the classification process and does not need to be a part of the feature extraction.

The signal correlation is one of the used features in [113] and [114]. In [56] the signal correlation between the x- and y- axis and the y- and z-axis of the accelerometer sensors resulted in one of the best classifications of the different analyzed feature combinations. The so called *weightlessness-based* features were compared to *traditional* features like the signal correlation in [70]. These *weightlessness-based* features are not applicable to the modular activity recognition proposed in this thesis, since activities need to be recognized that are distinguished through the orientation of the phone.

Fast Fourier Transformation (FFT) A feature extraction that is quiet different from those previously presented is the discrete Fourier transformation (DFT). A fast implementation with the same results as the DFT is the FFT, which transforms the input signal from the time into the frequency domain. Therefore, the FFT does not reduce the dimensionality and further feature extraction methods are needed to do that.

A Fourier transformation is, even with its FFT variant, very calculation intensive. Further, the dimensionality reduction which is needed afterwards does demand additional processing resources. Since one of the challenges of activity recognition on mobile phones is the limited resources, this feature extraction method is unfavorable. For the feature extraction on mobile phones the FFT is not well suited anyway, since the sampling of commercially available mobile phones offer rates in the 30Hz region. Usually a Fourier transformation needs window sizes higher than 256 samples, which are with the offered sampling rates over 8 seconds in length. Since many activities have higher frequencies in change, this method is only applicable when the windows are overlapping. As can be seen, the usage of this feature derives a lot of additional questions to be answered and variables to be determined. Since the focus of this thesis is the modular activity recognition and not the analysis of the best selection of features, the Fourier transformation is not considered in the reminder.

In [20] the FFT was used as basic feature extraction method with overlapping widows of 512 samples in size. Different features for activity recognition are analyzed in [80], which includes the FFT with different window lengths and overlapping.

Principal Component Analysis (PCA) The last feature extraction method presented here is not one per se, since only the dimensionality of the input is reduced to a desired length according to the principal components. The PCA is especially suited for the orthogonal transformation of highly correlated data, which would be the case for multi-axis accelerometers. The input is transformed according to the axis the data has the higher variance, which also results in a dimensionality reduction.

The PCA especially is used after a FFT to reduce the dimensionality of the feature vector to a classifiable length. Since in this thesis no FFT is used and the correlation between the different accelerometer axis is covered through covariant membership functions of the modular classification, the PCA is not considered as possible feature for the activity recognition.

A PCA was used in [99] to remove the correlation of the accelerometer axis and to reduce dimensionality. The approach was used for activity recognition to calculate the daily calorie expenditure of the user. The common principal components (CPC) identified through a PCA were used in [155] to project the feature vector to a lower dimensional space. In [96] a PCA was used in the feature extraction process.

Summary and Discussion Since the focus of this thesis lies on the modular activity recognition, which is the classification, the different feature extraction methods are not part of the evaluation. Therefore, features are chosen that provide the information which allows the recognition of all desired activities. In this manner two groups are

important, the static and dynamic activities. The mean feature allows the detection of the phones orientation, so it is suitable to recognize the static activities. Also, the effort to calculate it is the lowest of all previously described features. For detecting the dynamic activities the variance feature is sufficient and also efficient to be calculated, since it is combined with the mean feature.

Some of the features would bring new information such as the FFT, but due to other reasons this feature is not applicable to activity recognition on mobile phones. Most other features have overlappings with the features mean and variance, which is why they would not introduce relevant new information to the recognition process. There are newer more sophisticated feature extraction methods (like presented in [70]), which were not described in this subsection, but since the complexity of the modular activity recognition implements this features on the next processing level, these are not considered as related work.

2.2.2 Mapping Functions and Classifiers

For recognizing activities there is always some kind of classification process used. This process mostly includes a mapping function, which reduces the dimensionality to a one dimensional set and projects the feature vectors onto numerical class identifiers. After the mapping function the actual classification can be done. Sometimes the distinction between mapping and classification can not be made due to the nature of the classification algorithm. Different algorithms for classification in activity recognition and their appliances in healthcare, well-being and sports are again listed in [15]. In this subsection the algorithms most often used for activity recognition are introduced and systems are described which implement them. Applicability for the proposed modular activity recognition is outlined and if they are able to cope with the challenges of the recognition on mobile phones.

Artificial Neural Networks As mapping function of feature vector to classifiable one dimensional values often an artificial neural network (ANN) is used. An ANN consists of one or more layers of neurons which process their input and feed the output to the next layer of neurons. Since the research field of ANNs is very old [139], there are various well researched types of ANNs. A good introduction to ANNs can be found in [163].

An ANN was used by Khan et al. [86] to detect 15 physical activities with one triaxial accelerometer firmly attached to the users chest. The ANN type used is a standard feed forward one, which is trained with a supervised back-propagation algorithm. Different variations of perceptron numbers and layers were tested to determine the optimal combination. In [87] the same authors again used an ANN for activity recognition. There again a triaxial accelerometer was firmly mounted on the users chest. The feed-forward ANN was also trained with back-propagation to recognize four activities. Again in [85] these authors applied feed forward ANNs in combination with back-propagation to classify human activities. There a smart phone was used as sensing device.

A multi layer feed forward ANN was used by Yang et al. [156] to detect eight activities in a home setting. The overall recognition system was highly complex and utilized filters, different feature extraction methods, a preclassifier and a classifier for both static and dynamic activities. One triaxial accelerometer sensing device was mounted on the users dominant wrist. The approach had been evaluated with seven test subjects.

User activities in a hospital scenario were estimated with ANNs by Favela et al. [53]. There five to six activities were recognized with multi layer feed forward ANNs. In [55] different feature extraction and preprocessing techniques were analyzed on context recognition and which classification algorithms are suited to classify them. Different activities were recognized based on accelerometer data from a mobile phone with a multilayer perceptron ANN by Kwapisz et al. [93]. The results were compared with two other algorithms, where the ANN performed best.

The biggest disadvantage of ANNs is the interpretability of the mapping function, since the internal structure of the ANNs is not readable even for experts. Here hybrid architectures between ANNs and fuzzy systems help overcome this issue. Also, the behavior of an ANN with outliers and unexpected input is not predictable, which again can be solved through the combination with fuzzy systems. Another issue is that the mostly used back-propagation training, which mainly has a long runtime and can sometimes tend to over-fitting. The training algorithm proposed in this thesis is much more efficient than standard ANN and hybrid fuzzy systems training [81]. Another issue is, that a ANN is not very useful in deriving a reliability measure, since no fuzzy values are used in this kind of mapping. A reliability measure is especially useful when unreliable data needs to be separated from reliable to improve the overall statistical accuracy of the activity recognition. Generally an ANN can be used as mapping function in

the proposed modular activity recognition approach, but due to the disadvantages a fuzzy inference system (FIS) is preferred.

Support Vector Machines A very popular and precise classification algorithms are support vector machines (SVM) [144, 47]. SVMs divide data of two classes in such a way that the separation border has the maximum distance between both class members. Usually the borders are a set of linear functions positioned by the support vectors, but there are non linear extensions for SVMs for non linear separable data. Another method for dividing non linear separable data are SVMs with kernel functions [22, 32], which transfer the data set into a higher dimensional space, where the data is linear separable.

Human activities were recognized with a SVM classifier by Cho et al. [43]. There a firmly attached smart buckle equipped with accelerometer and visual sensors was used. Nine activities were recognized on base of the accelerometer sensor measurements in total, where a FFT and a signal correlation were used as feature extraction methods.

An activity recognition based on SVM in elderly care is presented in [129]. The triaxial accelerometer sensor was worn on the right side of the users belt where it was firmly attached. In total nine activities were detected. Among seven test subjects a recognition accuracy of over 90% was achieved.

In [70] an autoregressive (AR) model of time-series is presented for activity recognition. The AR coefficients were classified with a SVM. The one-versus-one strategy was chosen to solve the multi-class problem of SVMs. A recognition accuracy of over 90% could be reached for the detection of four activities. A discrete cosinus transformation combined with SVM classification is presented in [69]. Also four activities were recognized with over 90% accuracy.

SVMs would be very suited for modular activity recognition with modules which recognize only two classes, the activity class and everything else, but with modules which recognize more then two classes, the multi-class issue of SVMs needs to be solved [37]. Also, SVMs are very accurate in separating linear and non linear separable data, where only a few training data pairs are available and algorithms like KNN tend to over-fitting.

One of the biggest problems with SVMs are speed and size of the training and the classification [37]. This would be a huge limitation using SVMs for activity recognition on resource limited mobile phones. Another issue with SVM classifiers with kernel functions is the choice of the kernel [37]. For activity recognition with mobile phones, which are not firmly attached to any position, the supply of a reliability measure to improve the robustness of the classification process is very important as the evaluation will show. Due to the discreteness of the classification process provided by SVMs, deriving a reliability measure on each classification is hardly possible. A fuzzy extension of SVMs [97] could help overcome this issue. Further, SVMs are very different in structure than e.g. Fuzzy Inference Systems (FIS), which rely on a deterministic rule structure. This rule structure allows the individual *activation* or *deactivation* of each of the input dimension of each rule. This allows the adaption of the classifier without destroying its original capabilities as the evaluation will show. As can be seen so far this is not possible with SVM classifiers. Nevertheless, SVMs would be applicable in a modular activity recognition with the proposed architecture and other novelties, but due to the disadvantages, which would need an extensive adaptation of the original concept, fuzzy systems are preferred.

Probabilistic Models Probabilistic models for classification of sensor data are e.g. Bayesian networks. A Bayesian network is a directed acyclic graph with nodes of random variables and edges with conditionalities. Another class of probabilistic models for the classification of accelerometer data in activity recognition are Markov models. The most often applied Markov model is the variant with hidden states (HMM).

A hierarchical dynamic Bayesian network based system for activity recognition is proposed in [131]. The system combined accelerometer measurements with GPS coordinates to jointly recognize activities and environment of the user. The authors claim that their approach yields 10% absolute improvement over other systems. Since a hierarchical classifier always needs a lot of supervision by an expert to design the system and the accelerometer sensor is not processed separately, this approach is not comparable to the approach presented in this thesis.

A fuzzy Bayesian network for the continuous classification of accelerometer and physiological sensor data was used by Yang et al. [157]. The sensor device was firmly attached to one of the upper arms of the user. The combination of Bayesian networks and fuzzy logic allowed the authors to not only recognize dynamic activities, which produce time series of sensor measurements, but also static ones. Also, an algorithm was introduced to train

the fuzzy Bayesian networks. The combination of accelerometer and physical sensors helps boost the recognition rates over the single sensor classification accuracy. This is also the biggest difference to the approach presented in this thesis, since modern mobile phones do not come with physical sensors, nor is it firmly attached to any body position.

Different algorithms for classification in activity recognition are analyzed in [117] and [20]. In [117] a naive Bayes algorithm is compared with decision trees, KNN and SVM classifiers. A plurality voting combining all baseline classifiers resulted in the most accurate recognition. Also in [20] a naive Bayes approach is compared with a decision tree, a decision table and an instance based learning (IBL) classifier.

Activity recognition in an assembly task environment on accelerometer data with HMMs is presented in [149]. The accelerometer activity classification was combined by various methods with a classification on audio data to improve recognition rates. Activities in a surgical environment were detected with HMMs by Ahmadi et al. [9]. There 14 activities were detected via three firmly attached accelerometer devices on the surgeons belt and wrists.

HMMs are very good in the recognition of sensor data time series, but with static activities, there is no change to the sensor data patterns, so transitions in the model from one state to another can not be made. Since in this thesis the limitation to the recognition of only dynamic activities is not made, this classification approach is not applicable. Same applies for Bayesian network based approaches with no extensions. The fuzzy extension bresented in [157] would also allow the recognition of static activities. In this thesis a fuzzy inference system (FIS) is used solely, since its description is more clear and the recognition results are comparable - recognition rates are expected to be much better due to the novel improvements to FISs presented in this thesis.

Decision Trees A hierarchical approach to classification of sensor data are decision trees. The classification process starts at the root of the tree and proceeds to a leaf, which indicates the classification output. Each node on the path to a leaf includes a decision which path further to proceed. Normally decision trees only can classify linear separable datasets, but there are non linear extensions. Algorithms to learn decision trees are e.g. ID3 and C4.5.

A decision tree for activity recognition for health care monitoring was tested by Hong et al. [75]. The authors detected five human body states with three accelerometer devices firmly attached to a wrist, the hip and a upper leg of the test persons. The output of the accelerometer activity recognition were combined with the object interaction detected through a RFID reader in a glove. The results of the proposed system are compared with an upper limit due to a VCO_2 measurement device.

In [104] the proposed approach is compared to a J45 decision tree and a naive Bayes algorithm, where the proposed approach performs slightly better than the standard classifiers. Also in [117],[93] and [20] decision trees are compared with other algorithms. In [20] the C4.5 decision tree had the highest recognition accuracy. Also, the evaluation presented by Ravi et al. [117] showed very good results for the decision tree classification.

One advantage of decision trees is, that they are easy to understand and provide a certain interpretability. Another advantage of decision trees is, that even though their training could be calculative intensive, the classification is not. Further, the decision tree classification is immune to the presence of useless features, since they are just ignored [109]. This advantage is the biggest disadvantage of decision trees to be used in the modular activity recognition proposed in this thesis. One of the challenges of activity recognition on mobile phones is that it needs to be flexible to changes in the users behavior or environment, which leads to different sensor patterns. The proposed method is to select individually which feature is best for classifying the patterns, which would not be possible with decision trees. Also, the provision of a reliability measure which improves the robustness of the recognition is not possible with a crisp classifier as it is a decision tree. Nevertheless, a modular activity recognition would be possible with decision tree classifiers.

Fuzzy Systems h The theory of fuzzy sets is very old [158]. It was originally designed to have a rule system in which human knowledge can be modeled and is also interpretable by humans. In fuzzy set theory first the crisp set has to be defined, so the difference to a fuzzy set can be seen:

Definition 1 (*Crisp Sets*)

The crisp set A over the universe $\mathcal{U}$ is defined through its characteristic membership function μ_A

$$\mu_A : U \rightarrow \{0, 1\},$$

where $\mu_A(x) = 1$ if $x \in A$ and $\mu_A(x) = 0$ if $x \notin A$. The crisp set is described as

$$A = \{(x, \mu_A(x)) : x \in \mathcal{U}\}.$$

According to the definition of the crisp set, the fuzzy set is described as follows:

Definition 2 (*Fuzzy Sets*)

The fuzzy set $\tilde{A}$ over the universe $\mathcal{U}$ is defined through its fuzzy membership function $\mu_{\tilde{A}}$

$$\mu_{\tilde{A}} : U \rightarrow [0, 1],$$

where $\mu_{\tilde{A}}$ describes the degree of membership to the fuzzy set. The fuzzy set is described as

$$\tilde{A} = \{(x, \mu_{\tilde{A}}(x)) : x \in \mathcal{U}\}.$$

With fuzzy sets there is no binary decision to be made for whether or not a value belongs to a set, which is especially useful when describing activity classes in activity recognition. There classifications are usually made whether the feature vector input is assignable or not. Is the activity class described in a fuzzy way, the reliability of the current classification can be expressed. This is especially useful when reliable and unreliable classifications are separated to improve the overall statistical accuracy therefore providing robustness of the activity recognition.

Classification methods that involve fuzzy set theory are fuzzy inference systems (FIS) (fig. 4). In a FIS the input, such as a feature vector, is first fuzzified so the fuzzy inference can classify the input. The fuzzification is done according to fuzzy membership functions provided through values in the knowledge base. The rules for the

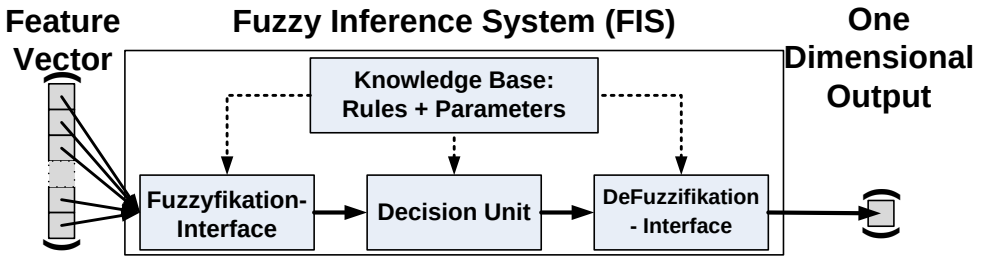


Figure 4: Basic structure of fuzzy inference systems (FIS).

class inference are also saved in the knowledge base. Each rule consists of an antecedent and a consequence part. Furthermore, the antecedent consists of membership functions used for fuzzification. The output of the inference is a fuzzy value, which has to be de-fuzzified so a crisp classifiable value is the outcome. Also, the parameters for de-fuzzification are stored in the knowledge base. If a reliability measure needs to be deduced, the de-fuzzification is obsolete and the inferred fuzzy value can directly be interpreted as class and reliability measure.

There are special FISs which are especially suited to be trained by machine learning. One is the Takagi, Sugeno and Kang [136, 132](TSK) fuzzy inference system (FIS), which schematically is described in figure 5. With a TSK-FIS the inference in the decision unit and the de-fuzzifikation is one indivisible process. Therefore, the output of the TSK-FIS can not directly be interpreted as class and reliability measure, but with an additional fuzzy classification the TSK-FIS output can again be fuzzified. A machine learning to train such a TSK-FIS called Adaptive Network-based Fuzzy Inference System (ANFIS) was introduced by Jang et al. [81], where in this thesis a different novel

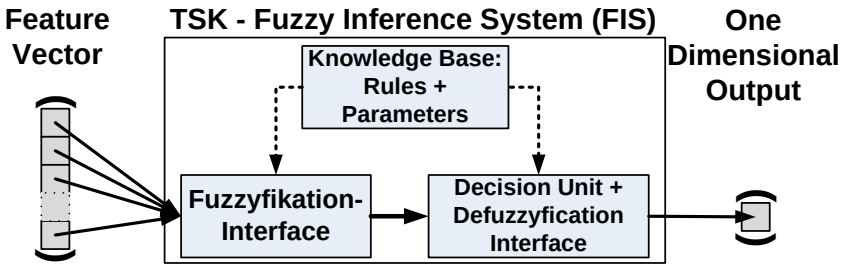


Figure 5: Basic structure of Takagi, Sugeno and Kang fuzzy inference systems (TSK-FIS).

machine learning algorithm is introduced. This novel training is lower in computational complexity and more accurate in case of highly correlated accelerometer features than the machine learning due to ANFIS.

Activity recognition in combination with fuzzy inference systems (FIS) was described previously by Helmi et al. [73]. The authors recognized four activities with one triaxial accelerometer firmly attached to the users waist. The fuzzy rules were used for part of the feature extraction and the feature selection. The proposed fuzzy rules were not trained with any machine learning, but constructed manually according to observations. Since with increasing number of activities to be recognized the effort of constructing such fuzzy rules is increasing and such a manual method needs expert knowledge, which can not be provided by a common user, thats why a machine learning approach is favored in this thesis.

In [155] a neuro-fuzzy classifier with feature reduction is proposed. For reducing the features to the relevant ones a feature subset selection and a linear discriminate analysis (LDA) was used. The neuro-fuzzy classifier was identified with a modified mapping-constrained agglomerative clustering algorithm upon training data from a accelerometer. The accelerometer sensing device was mounted on the users dominant wrist. The authors evaluated their approach by recognizing eight activities with seven subjects. The proposed approach is not very different from the one proposed in this thesis, but in [155] a standard TSK-FIS is trained. Further, the clustering algorithms used in this thesis provide covariant membership functions, which are suited for highly correlated accelerometer data. The feature selection is done in this thesis on demand to cope with the challenge of **flexibility**.

Fuzzy systems are mostly used for sensor data fusion such as in [102]. Since the fusion of different sensor sources is not the focus of this thesis, where as single source the mobile phone is used, this topic is not further discussed or elaborated.

Summary and Discussion As described with the classification algorithms and mapping functions introduced in this subsection, each method has its advantages and disadvantages, but one method has the most advantages and is providing the best capabilities to cope with the identified challenges. Therefore, in this thesis as mapping function a TSK-FIS is used in the modular activity recognition. It outperforms most of the other techniques and offers possibilities to cope with the challenges of activity recognition on mobile phones.

The biggest disadvantage with ANNs like these used in [86] and [156] are that they are mostly black boxes which do not allow any interpretation of the internal structure. Fuzzy systems are especially designed to overcome this issue. Also, the mapping of outliers and unexpected input is unpredictable, which again is not the case with fuzzy systems. Outliers and unexpected input are not covered through the membership functions and therefore their membership to the classification is very low or even zero. Interpreted as reliability measure this membership helps to sort this classifications out. Finally, a reliability measure can not sufficiently be derived with ANNs, since they provide no internal uncertainty. A filtering upon this reliability measure is the real strength of the fuzzy systems, since it increases the overall statistical accuracy and improves the **robustness** of the classification.

For SVM, which are e.g. used in [43] and [129], the biggest drawback is the calculation effort and memory consumption of both, the training and the classification. Whereas the training can be done on a server, the classification itself should run on a mobile phone, which lacks both, a fast processor and sufficient memory. Further, processor and memory need to be powered through a battery, which will therefore decrease in runtime. Other issues can partly be solved due to extensions, but the power consumption due to the SVM classification is a killer argument for the

usage on mobile phones. Nevertheless, SVMs are especially suited for modular classification, where only one class is distinguished from all the others.

Probabilistic methods like naive Bayes and HMMs are used e.g. in [117] and [149]. Since HMMs are only capable of recognizing activities which provide different time series of sensor data and in this thesis also the recognition of static activities should be possible, the usage of HMMs is not adequate. Same applies for Bayesian methods.

With the often used decision trees, e.g. in [75], the biggest advantage is that they are immune to useless features. Exactly this advantage is a disadvantage in this case, since the **flexibility** of the modular activity recognition can (among others) be guaranteed through the individual switching of features in the fuzzy rules. This offers an adaption of the classifier modules without destroying their original mapping capabilities. Also, this adaption technique can be removed any time. Further, a reliability measure can not be derived with decision trees, since they do not implement the fuzziness which is needed for that.

As mentioned before, the derivation of a reliability measure is possible with fuzzy systems, which especially offers **robustness**. Also, the membership functions used in in this thesis are especially suited for mapping highly correlated accelerometer data. Another advantage is the interpretability of a TSK-FIS, where each rule can clearly be assigned to the class it should map onto. Each fuzzy rule has also the same length and therefore the calculation effort can easily be determined. In general the effort for calculating a TSK-FIS is better than most other methods and worse than some, but the biggest advantage lies in the **flexibility**. In a TSK-FIS each input of each rule can individually be activated or deactivated during runtime, which makes the mapping function adaptable without loosing its original mapping capabilities.

2.3 Modular Classification

The key contribution in this thesis and a relatively new concept in the field of activity recognition is the modularity of the sensor data classifier. In other application fields, the fusion of classifier modules has been investigated for many years now and in the research field of activity recognition the combination of classifiers becomes more and more relevant, but the modularity presented in this thesis is novel and specially developed for activity recognition on mobile phones.

In general [152], two approaches are differentiated: fusion on the abstract level and on the measurement level. A more fine grained overview of classifier fusion methods is listed and described in [121]. This subsection is structured accordingly.

First, classifier fusion on soft-output is introduced. There are several methods for soft-output fusion, where here the most related ones are presented along their implementations, the fuzzy and Bayesian fusion methods. Second, classifier fusion methods according to class ranking techniques are listed along with systems implementing this methods. The class ranking methods are subdivided into class set reduction and class set recording methods. Third, single class label classifier fusion is explained, also along systems implementing the methods. Voting and behavior-knowledge space methods are the subcategories in this case. Fourth, methods operating on classifiers are listed. The subcategories in this case are a hierarchical mixture of experts and classifier selection methods. The classifier selection methods come closest to the novel approach presented in this thesis. Last, all the approaches are summarized and compared. The differences and similarities to the proposed method are listed and discussed.

2.3.1 Soft-Output Classifier Fusion Methods

According to [121] the largest group of classifier fusion methods are the ones on soft-output. The fusion is done according to fuzzy measures such as probability[1], possibility[1], necessity[1], belief[1] and plausibility[1]. In this manner this kind of fusion method tries to reduce the uncertainty. Two main research fields in this group are introduced in this subsection.

Bayesian Fusion Methods Very popular fusion methods on soft classifier outputs are in general probabilistic and in particular Bayesian. There the posterior probability of the classification, a value in the range of $[0, 1]$, is used to combine the single classifier outputs. This results in a more fine grained fusion result, where not only the crisp class determines the outcome, but also the probability.

In [76] a multiple classifier fusion called Bayes-based confidence measure is compared towards ordinary Bayesian fusion. The authors claim that their approach leads to reductions of 20% and 25% in EER[2] and ROC[3] curve area. The approach was validated along the application of speaker verification.

Another classifier combination based on Bayesian methods is proposed in [12]. The authors investigated if there is a set of dependent classifiers which result in a better combination than independent classifiers with naive Bayes fusion. A non-Bayesian probabilistic classifier fusion method is presented in [138]. The proposed approach was compared to other popular classifier fusion approaches on a synthetic dataset and two datasets with symbols and digits.

A naive Bayesian fusion was also used by Zappi et al. [162], but in this case in the research field of activity recognition. The authors detect activities in a car assembly environment.

Fuzzy Fusion Methods Another group of fusion methods on soft-output is the fuzzy group, where the classifiers deliver a measure in the range $[0, 1]$. There are many differences between fuzziness and probability [67], but in the end they express the same uncertainty of the classification process and are complementary rather than competitive [159]. This uncertainty helps to improve the fusion outcome towards a crisp class combination.

Fuzzy rule based classifiers were combined by Assareh et al. [13]. The authors used a hybrid random subspace fusion scheme to combine the classifiers. Two data sets of cancer protein mass spectra were used for evaluation of the proposed approach.

[1] AI terms for non crisp uncertain classification information for different levels of processing and with different information content.
[2] Equal Error Rate (EER): Describes the intersection of the ROC curve with the diagonal.
[3] Receiver Operator Characteristic (ROC): Method for evaluation and optimization of analysis strategies [5].

In [64] a multi-classifier fusion based on fuzzy clustering is proposed. The approach was tested on a synthetic 2D dataset. Another approach for multi-classifier fusion using fuzzy templates is presented in [90]. The approach was evaluated on data sets from the ELENA database [2].

2.3.2 Class Ranking Based Techniques

According to [121] there are two methods for classifier fusion by class ranking. One is reducing the set of classes from the classifier outputs to a minimum, but ensuring that the correct class is still in the set. The other one tries to improve the overall rank of the true class.

Class Set Reduction Methods A reduction of the class set is one technique to fuse classifiers outputs via ranking. Two criteria have to be considered when reducing the set, the size of the set and the possibility of the correct class being still part of the set. This method tries to find the best tradeoff between both.

A class set reduction strategy according to the neighborhood intersection is proposed in [111]. The proposed approach was used to detect pedestrians and vehicles in a cybercar environment. For classifier fusion a Mamdani fuzzy system was used.

Class Set Recording Methods The class set recording methods try to improve the overall rank of the correct class. Three different methods are distinguished, the highest rank, the borda count and the logistic regression method.

Several methods for a class set recording fusion were evaluated by Ward et al. [149]. The aim was to recognize activities in an an assembly and maintenance scenario, which is the closest to this thesis's aim so far. The accelerometer and sound classifications were fused with the highest rank method, the borda count method and logistic regression. A simple comparison of the two sensors classifiers performed mostly better than the other techniques.

Another publication that comes close to the application field tackled in this thesis is [78]. A collaborative context recognition for mobile devices in which classifier fusion methods like borda count, majority voting, weighted majority voting and a binary protocol were used. The approach had been evaluated in an activity recognition scenario, where the user wears a heart rate sensor, a accelerometer sensor on the hip and on the wrist. The classifications of one to six subjects were fused with the proposed voting methods, where the best results were reached by the weighted majority voting.

2.3.3 Fusing Single Class Labels

Another group of fusion methods proposed in [121] are the fusion methods on single class labels. The crisp classification output of the classifiers is combined with voting or behavior-knowledge space methods.

Voting Methods Very simple methods to fuse crisp classifier outputs are voting methods. There the multiple classifiers outputs are combined in different ways e.g. according to the majority of equal labels. These methods are usually performing less accurate than methods operating on soft classifier outputs.

Huuskonen et al. [78] used voting methods on single class labels to combine classifier outputs. The fusion methods majority voting and weighted majority voting were compared to other fusion methods, and the majority voting was performing best.

Zappi et al. [162] compared majority voting as classifier fusion method with a naive Bayes combination. The authors tried to detect 10 activities in a car assembly environments. The amount of accelerometer sensors firmly attached to the test person and therefore the amount of classifiers was constantly improved to increase recognition accuracy. As fusion method the naive Bayes outperformed the majority voting.

Reputation-based voting and majority voting were used by Bahrepour et al. [16] to fuse different sensor sources in a medical monitoring environment. The authors firmly attached five triaxial accelerometer and gyroscope sensor nodes to test persons to recognize the activities.

Behavior-Knowledge Space Methods With behavior-knowledge space (BKS) methods, the independence of the single classifiers output is not assumed. There the knowledge of all decisions of classifiers is combined to provide a more accurate fusion.

A BKS fusion method for indexing continuous video content is proposed in [130]. The authors adapted the usual BKS method for continuous data. A BKS method for classifier fusion is used in [46] for network intrusion detection. The BKS method was analyzed according to generalization errors and strategies for performance improvement by Sarunas et al. [116].

2.3.4 Methods Operating on Classifiers

A group of classifier fusion methods operate on the classifiers itself rather than on the classifiers output [121]. The dominant role in this group is played by the dynamic classifier selection, which also comes closest to the approach presented in this thesis.

Hierarchical Mixture of Experts One method of organizing classifiers is hierarchical, where each sub-classifier has its unique position. The output of one level of classifiers is mostly fused by the next layer until a final decision is made about the overall output.

As a hierarchical mixture of experts based on the divide-and-conquer principle the approach presented in [82] can be seen. The authors used a triaxial accelerometer attached to the users hip to recognize 12 tasks for ambulatory monitoring.

Other hierarchical methods of organizing classifiers applied to activity recognition are shown in [19] and [107]. In [19] classifiers outputs were firstly combined on class level, than a further output fusion is done on the source level and finally on the method level the overall output is derived. Activity monitoring for the elderly was proposed by Muscillo et al. [107], where a hierarchical classifier based on Dynamic Time Warping (DTW) is used.

Dynamic Classifier Selection Another group of methods which operate on classifiers are dynamic classifier selection methods. As the name suggests, through a scheme the single best classifier is dynamically selected according to the respective input or output.

A dynamic sensor selection scheme for activity recognition is proposed in [160] in combination with a fusion of the individual classifier outputs by a meta-classifier. There the selection scheme was investigated to efficiently use the provided limited energy while achieving desired accuracy. The contribution of the respective sensor classifier to the overall recognition accuracy was determined during system training. The authors used various sensor devices firmly attached to the users arms to detect ten activities.

Other dynamic classifier selection schemes not applied to activity recognition can be found in [59] and [166]. In [59] two methods for dynamic classifier selection are proposed. The methods were evaluated on data sets from the ELENA database [2]. A dynamic classifier selection for effective mining from noisy data streams is proposed in [166].

2.3.5 Summary and Discussion

A broad variety of classifier selection, combination and fusion method have been presented in this subsection. Many methods have not been used for activity recognition up to this point, but some were. Approaches which propose a fusion of classifier outputs by class rankings are [149] and [78]. Since in this thesis the output of classifiers is not fused, but the classifier modules activation is of concern, these methods are not of further relevance. Also, in [149] the fusion of outputs of classifiers from different type of sensors are described, where in this thesis only the accelerometer sensor is of concern.

Nevertheless, a fusion method on soft classifier outputs of two different sensors using fuzziness is proposed in [23], in which part of the applied methods of this thesis are used. A system that also applies a soft classifier output fusion is presented in [162], but their case was based on probabilistic uncertainty with a naive Bayes method.

The group of methods that comes closest to the novel modular activity recognition proposed in this thesis are the methods operating on classifiers them self. There the dynamic classifier selection is the most related one (e.g. [160]), but instead of a mechanism that selects the classifier module to be active, the modules schedule themself. This kind of modular classification method had not been used in any other system so far, especially not for activity recognition on mobile devices, where its real strength is. In this thesis the possibilities of such an self scheduling modular classification with reliability is evaluated in detail on five challenges which have been identified for activity recognition on mobile phones.

Compared to the dynamic classifier selection methods presented in [59], [166] and [160], the proposed method is more **flexible**, since no additional selection function has to be designed, trained or readjusted. This would especially be the case if classifier modules are adapted or exchanged. Such a selection function would also need certain meta knowledge of the single classifier, but with the modular activity recognition proposed in this thesis the classifiers are considered as black box and select themself. Also, hierarchical mixtures of classifiers as presented in [82], [19] and [107] are very **inflexible**, since usually a lot of expert knowledge is needed to define the hierarchy and the structure needs to be redesigned on every classifier which needs to be adapted or exchanged. Additionally, a dynamic classifier selection and hierarchical mixtures of classifiers prevent the activity recognition from being **extensible** since the selection function needs to be changed or the hierarchy be redesigned. The proposed modular activity recognition provides **extensibility**. Another challenge which can be coped with due to the modular approach proposed in this thesis is the reaction on different **conditionalities**. This would also be possible with classifier selection or hierarchical classifiers, but the complexity of such a system would rise and therefore would increase the **resource** consumption, which is not the case with the novel modular activity recognition.

3 Modular Activity Recognition Architecture

With the design of an activity recognition system for mobile phones various challenges need to be mastered. Providing a method to cope with each of the challenges **flexibility**[1], **extensibility**[1], **robustness**[1], **resources**[1] and **conditionality**[1] is ambitious, but for all seems to be hardly possible. Nevertheless, for this thesis in general and in particularly this section, an architecture for activity recognition on mobile phones is presented that can deal with all five challenges together. Although some of the challenges are partly contradictory, such as **resources** and **robustness**, the proposed modular activity recognition delivers good tradeoffs.

The modularity of the activity recognition is the key innovation presented in this thesis, but other improvements to the classification process and the machine learning deliver approaches to cope with the challenges, too. To deal with the challenges of **flexibility** and **extensibility** a mapping function in the classification process is needed which can be adapted individually and dynamically onto the changes without destroying the original mapping capabilities. A **robust** activity recognition based on accelerometer sensor data needs to be especially suited for highly correlated data. Also, to deliver real **robustness** the classification somehow has to be self-stabilizing and provide a reliability measure to sort out incorrect recognitions. Dealing with many different **conditionalities** of activities mostly results in an increased complexity of the recognition, but this usually would lead to a rise in **resource** consumption. The limitation of the **resources** available is obvious anyway for the chosen platform mobile phone.

In this section the architecture of the novel activity recognition for mobile phones is described. A graphical overview about this section is given in figure 6.

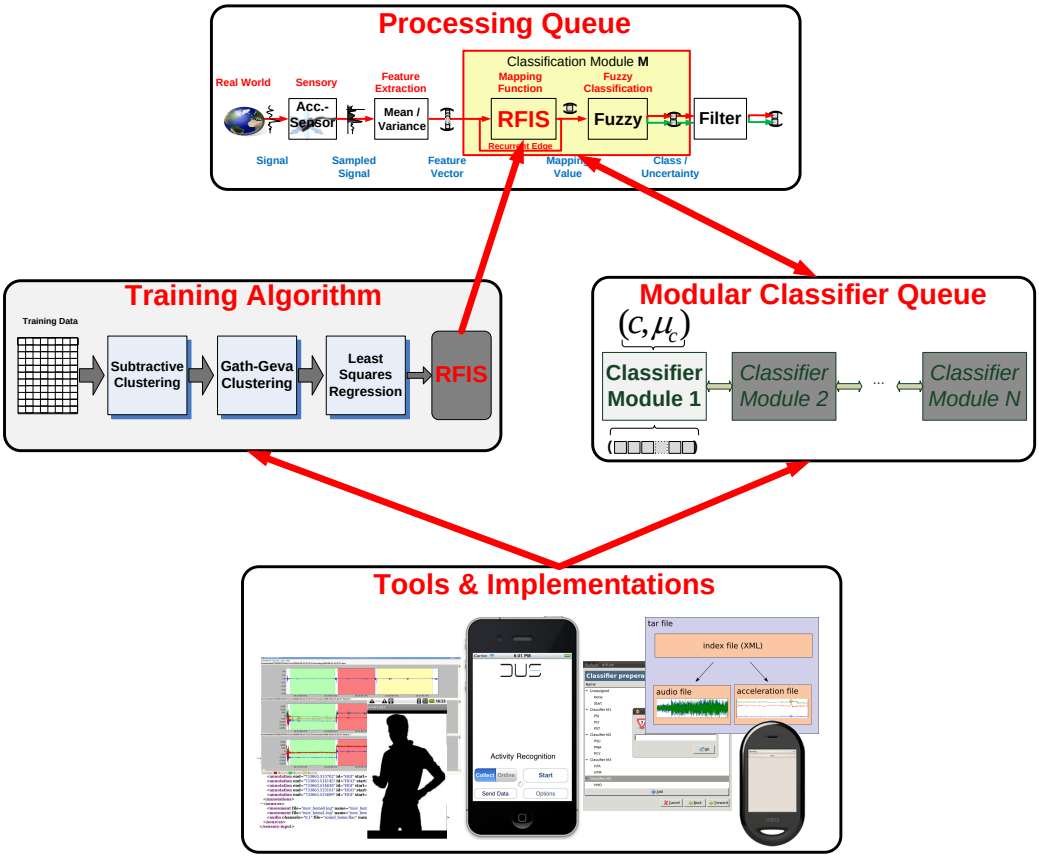


Figure 6: Modular activity recognition architecture - processing queue, modular classification, training algorithm, tools and implementations.

[1]The bold words correspond to the five challenges of activity recognition on mobile phones: **flexibility**, **extensibility**, **robustness**, **resources**, and **conditionality**

First, the queue of processing from real world signal to filtered class identifier is described. The processing starts with the sampling of the accelerometer sensor measurements. Then the windowing of the sensor data stream and the feature extraction is defined. The key component of the processing queue is the mapping function, where a novel recurrent fuzzy inference system (RFIS) with covariant membership functions is used. The actual classification is the next step of processing, which is done according to fuzzy numbers that deliver not only a crisp class, but also a reliability measure. The last step of processing is a filtering upon this reliability measure, which can be influenced or even switched off due to a threshold. Additionally a bit-vector masking for the mapping function is defined, that enables the adaption of the classification without destroying the original capabilities.

Second, the modularity of the activity recognition and it's key improvement is described. As module the composition of the RFIS mapping function and fuzzy classification is defined. Instead of only one module doing the activity recognition many modules are responsible. The module switch is accomplished due to the recognition of a so called complementary class. This complementary class demands the adjustment of the fuzzy classification. The classification modules can be arranged in various ways, where the only researched one so far is the dynamic queue. A probabilistic arrangement and other techniques are suggested as well.

Third, the novel training algorithm for the RFIS mapping function of the classification modules is defined. Since the mapping function includes a recurrent edge and covariant membership functions no standard machine learning like that proposed in [81] can be used. Also, having many modules for recognition instead of one monolithic classifier recommends special considerations as well. Therefore, a novel training algorithm is described, which involves different clustering techniques, genetic algorithm search and linear regression. Additionally the genetic algorithm to identify bit-vectors for the module masking is explained.

Lastly, the implementations of the activity recognition, the training algorithms and the tools supporting the activity recognition are described. Since the aim of the novel modular activity recognition is not only the proof of concept, but also the deployment on mobile phones so common users can profit of it, three mobile phone platforms are supported. Having only the implementations of the activity recognition and the training algorithm is not enough for systems based on sensor data. A tool chain is needed, which supports the sensor data collection, the data annotation, and the editing. Also, formats for the sensor data and the classifier specification need to be defined such that each tool or implementation is connected seamlessly.

3.1 The Processing Queue

The classification consists of several steps of processing a real world value to a tuple of the class recognized and a fuzzy reliability value. The processing queue is shown in figure 7 and described in the following.

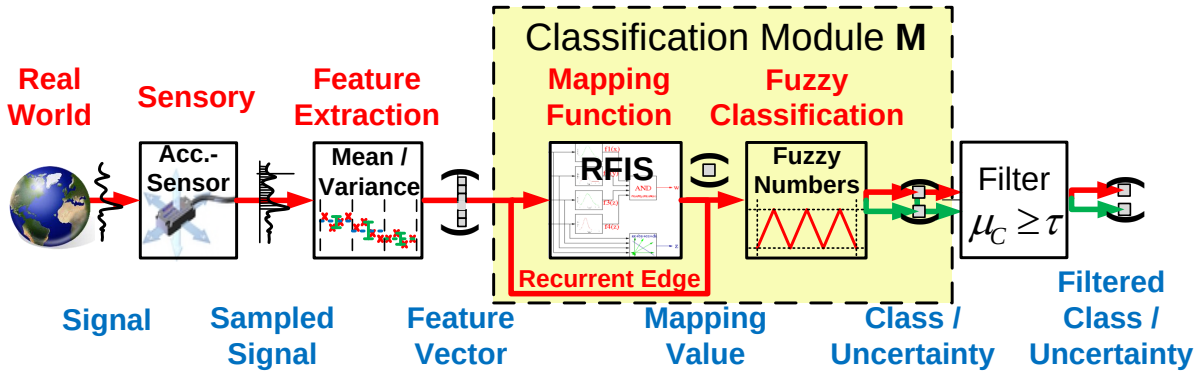


Figure 7: Classification system architecture.

First, the sensor has to measure the real world value. In this case the sensor is one or more accelerometers with 3-axes. The sensor measurements are streams of digitalized data, which are numbered according to the time of occurrence.

Second, the feature extraction, which is used for dimensionality reduction and extraction of features indicating the activities, is explained. The feature extraction process composes of windowing the sensor streams, mean extraction, variance value calculation and composition of the feature vector. Also, the used features are motivated by how they are used and their benefits over other features.

Third, the used mapping function, a Recurrent Fuzzy Inference System (RFIS), is deducted from general Fuzzy Inference Systems (FIS). The mapping function maps the feature vector onto a classifiable one-dimensional set. In particular a Takagi, Sugeno and Kang (TSK)-FIS with covariant membership functions (MF) is used, which is especially suited for automated construction. Also, the feedback of the output as additional input to achieve a recurrent mapping function is defined. The modularity and other improvements introduced in this thesis are applicable to other mapping functions as well.

Fourth, the fuzzy classification is defined, which results in a tuple of class and fuzzy reliability measure. Input of the classification is the output of the mapping function. The classification is done according to fuzzy numbers, which are also defined in this section. A classification with fuzzy numbers in combination with the recurrence of the mapping function delivers a reliability measure, which describes the confidence of the classification process.

Fifth, a filtering upon the reliability measure is described, which improves the overall accuracy. Input of the filtering is the tuple of class and reliability measure, which are the output of the fuzzy classification. The filter excludes classifications from further usage which are unreliable according to a threshold. Also, the determination of the filter threshold is explained.

Last, a bit vector masking is introduced, which is used for personalization, generalization and adaption of the mapping functions. This step is additional to the processing queue, because it modifies the mapping function temporarily and is therefore only indirectly part of the processing queue.

3.1.1 Accelerometer Sensor Sampling

For the activity recognition presented in this thesis only accelerometer sensors are used. Most of the nowadays mobile phones have one 3-axes accelerometers built in, some have more. The sensors of the mobile phones are readout through a micro-controller. The sensor measurements are provided through an Application Programming Interface (API) which hides the underlying sensor driver. Also, the API mostly does not allow many configuration possibilities. Therefore, the offered sampling rates are in the range of $30 - 100Hz$, which are on the lower end of sampling frequencies for activity recognition [35]. Vibrations can not sufficiently be detected with this sampling

rates, which makes the usage of a Fourier Transformation (FT) as feature extraction method unfunctional. The usage of the offered API is however unremitting. If a sensor driver would be implemented for better performance, the hardware or operating systems vendors would not support this for the common user. So, in this thesis the evaluations are done only on what is offered through the implemented API, since the focus is every user and not only researchers.

Sensor Data Stream Each of the accelerometer sensors has three axes, which are processed separately. The sensor data streams of each axes are shown in the following:

$$x_{k,1}, x_{k,2}, .., x_{k,t}, x_{k,t+1}, ..$$

$$y_{k,1}, y_{k,2}, .., y_{k,t}, y_{k,t+1}, ..$$

$$z_{k,1}, z_{k,2}, .., z_{k,t}, z_{k,t+1}, ..$$

The sensor data stream starts at $t = 1$ with $t \in \mathbb{Z}^+$ and is considered to be infinite. In practice the mobile phone, the activity recognition application or the sensor gets switched of at some point, but this time is not known in advance and therefore no boundary is assumed. The time points t are no real time stamps rather than an integer numbering of the occurred measurements. In reality time points t could even be non-equidistant measurements, which results out of the nature of the sensor sampling and buffering.

The sensor data streams of each axes and each sensor are rationed into windows of further processable data sets, which is described in the following.

3.1.2 Mean and Variance Feature Extraction

Out of the variety of feature extraction methods listed in subsection 2.2.1 a few had to be picked, leading to good activity recognition results. Since the focus of this thesis lies not on the feature extraction, but the mapping function and the classification itself, the most practical and expressive feature extraction methods are used. Also, if the mapping function is complex and can adapt onto non-linear data, the feature extraction can be held simple.

Sensor Data Windowing For processing the continuous raw accelerometer sensor measurements stream the window (see example in fig. 8) needs to be determined, which encloses the amount of data that is processed in one feature extraction step and therefore leads to one classifiable feature vector. The window size w strongly depends on the sampling rate of the sensor, which varies in most of the mobile phone models. In this thesis the window size w is not previously determined according to the analysis of the nature of the activities, but was decided on according to experience.

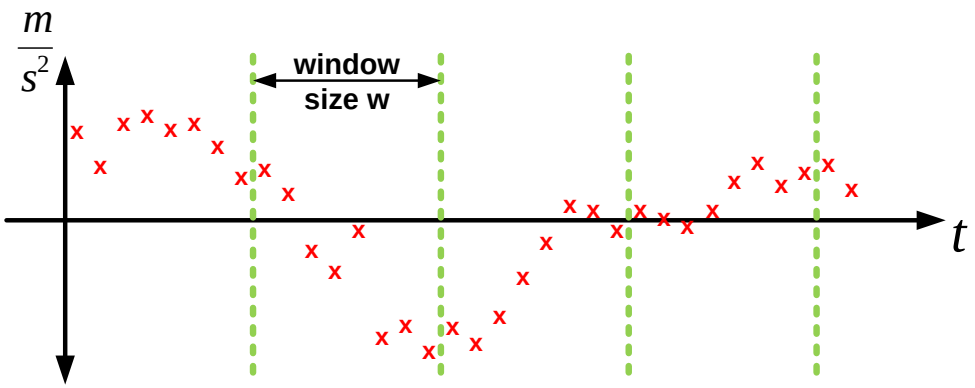


Figure 8: Exemplary windowing of accelerometer sensor measurements stream.

Furthermore, the windows are chosen to be non-overlapping, which seemed to be the right choice to limit calculation time for the activity recognition. If the windows would be overlapping a processing overhead for already

processed sensor data is needed. The used recurrent classification process includes awareness of the previous feature vector, which compensates the non overlapping windows. Also, the size of the processing window w determines the calculation effort for the whole processing queue. The smaller the window the more classifications have to be calculated per second.

Each window for each of the 3-axes of every accelerometer sensor is defined by vectors $\overrightarrow{x}_{k,t}$, $\overrightarrow{y}_{k,t}$ and $\overrightarrow{z}_{k,t}$ in the following:

$$\begin{aligned} \overrightarrow{x}_{k,t} &= \left(x_{k,t}, x_{k,t+1}, .., x_{k,t+(w-1)}\right) \\ \overrightarrow{y}_{k,t} &= \left(x_{k,t}, x_{k,t+1}, .., x_{k,t+(w-1)}\right) \\ \overrightarrow{z}_{k,t} &= \left(x_{k,t}, x_{k,t+1}, .., x_{k,t+(w-1)}\right) \end{aligned} \tag{1}$$

The window starts with sensor measurement at time t and ends at $t+(w-1)$, which results in a vector of length w. The index $k = 1, .., l$ denotes the respective sensor, since some mobile phones have more than just one accelerometer sensor.

Mean To determine orientation dependent activities where nearly no movement is performed, a feature is needed to extract the direction of the mobile phone. Since gravity always causes an acceleration, the used 3-axes accelerometers provide information about the orientation of the device. The feature extraction of a mean value generally does not destroy this information when no other high accelerated movements occur. Also, the calculation time for the extraction of the mean feature is really low, since only the individual measurements need to be summed up and then be divided through a constant.

The feature extraction method mean is described in the following equation according to the sensor measurements $x_t, .., x_{t+(w-1)}$ for one calculation window of the length w:

$$m(\overrightarrow{x}_{1,t}) = m(x_{1,t}, .., x_{1,t+(w-1)}) = \frac{1}{w}\sum_{i=0}^{w-1} x_{1,t+i} \tag{2}$$

The feature extraction window therefore starts at time t and ends at $t+(w-1)$ which includes w accelerometer measurement values. Each of the three axes of the sensor is processed separately.

Variance Another feature is needed to enable the recognition of activities with medium to high movement. The most simple is the variance of the sensor measurements in a time window. This feature has the advantage over e.g. the standard deviation that no complex mathematical operations like the square root are required. Also, the result from the mean calculation (eqn. 2) can be reused, which leaves only w subtractions, w additions and a division.

The variance is calculated with the inputs $x_t, .., x_{t+(w-1)}$ of the window size w according to the following equation:

$$\sigma^2(\overrightarrow{x}_{1,t}) = \sigma^2(x_{1,t}, .., x_{1,t+(w-1)}) = \frac{1}{w}\sum_{i=0}^{w-1} \left(m(\overrightarrow{x}_{1,t}) - x_{1,t+i}\right)^2 \tag{3}$$

Again, all of the three axes of the accelerometer sensors are processed separately.

Feature Vector The feature vector is the input of the mapping and therefore its dimensionality influences the complexity of the function. This circumstance influenced the decision on the usage of only two features in the processing queue. The order of the features in the feature vector is arbitrary and just has to be chosen when the mapping function is trained. How the feature vector is composed can then be specified in a configuration file, which is described in subsection 3.4.2.

The feature vector is exemplary defined for one 3-axes accelerometer sensor in the following:

$$\begin{aligned} \overrightarrow{\mathbf{v}}_t &= (v_{1,t}, v_{2,t}, .., v_{n,t}) \quad \text{, with } n = 6 \\ &= \left(m(\overrightarrow{x}_{1,t}), m(\overrightarrow{y}_{1,t}), m(\overrightarrow{z}_{1,t}), \sigma^2(\overrightarrow{x}_{1,t}), \sigma^2(\overrightarrow{y}_{1,t}), \sigma^2(\overrightarrow{z}_{1,t})\right) \end{aligned} \tag{4}$$

The example feature vector $\overrightarrow{\mathbf{v}}_t$ has six dimensions, since the features of only one 3-axes accelerometer sensor are used. If a mobile phone is equipped with more than just one sensor, the feature vectors dimensionality n rises in the manner $n = s \cdot 6$, where s is the quantity of sensors.

This feature vector is input of the next chain in the queue, the mapping function. Since the mapping function uses itself covariant membership functions (MF), features like signal correlation are not needed. Also, the high complexity of the mapping function used in this thesis compensates some of the other features left out the vector as well.

3.1.3 Recurrent Fuzzy Inference System (RFIS) Mapping Function

As described in 2.2.2, there are various ways of mapping feature vectors onto classes or classifiable linear sets. In this thesis a Fuzzy Inference System (FIS) is used. There are many reasons for this choice, but the methods described here can be applied to most of the previously described mapping functions and classification algorithms.

Arguments for using a Fuzzy Inference System (FIS) One reason for using a FIS in sensor data processing for activity recognition is the human ability to interpret the fuzzy rules. Fuzzy Logic was especially designed to enable humans to model their knowledge [158]. Even though the fuzzy rules in the used FIS are identified through a machine learning algorithm, the expert can read and interpret them.

Another reason for preferring a FIS as mapping function above (e.g. Neural Networks (NN)) is that the fuzzy models are closed. For a feature vector it always can be decided if it can be mapped by the FIS or if it lies outside the fuzzy model. To determine the crisp membership of the feature vector to the fuzzy model alpha cuts [158] over all membership functions of all rules need to be made.

Also, with fuzzy sets there are membership functions (MF) used, which enable the calculation of a reliability measure. The MFs in the FIS not only deliver a dimensionality reduction in the mapping process, but also produce a degree of belongingness of the input value to the classification. This is especially the case when the output of the FIS is fed back as additional input. More on this is described in subsection 4.3 and in [23].

A very important possibility with fuzzy rule based mapping functions is to offer **flexibility**. For every rule each of the feature vectors dimension can individually be activated or deactivated. Therefore, an adaption, personalization and generalization of the classification process is offered which does not destroy the original capabilities and can be removed at any point. This is the strongest advantage above e.g. SVM classification. The technical aspect of this so called bit-vector masking is described in 3.1.6 and the capabilities are especially analyzed in subsection 4.1 and 4.2.

The last reason to be named here is the easily determinable and deterministic effort for calculating the FIS. All the rules used in the mapping of the FIS have the same size and therefore need the same calculation time.

There are many more reasons to use FISs, but the choice here is arbitrary. As mentioned before, the modularity and other aspects of the proposed novel activity recognition architecture could be applied to most of the other mapping functions and algorithms, too.

Multivariate Takagi, Sugeno and Kang (TSK) - FIS In this thesis a Takagi, Sugeno and Kang [136][132] (TSK)-FIS is used, which is especially suited for automated construction [81]. A TSK-FIS is used to map the feature vectors onto a linear set, whose values can be assigned to a class identifier in a separate classification process. Within a TSK-FIS, the consequence of each rule is not a functional membership to a fuzzy set, but a constant or linear function. The consequence of the rule j depends on the input vector $\overrightarrow{\mathbf{v}}_t$ of the FIS:

$$\begin{aligned} f_j(\overrightarrow{\mathbf{v}}_t) &:= a_{1j}v_1 + .. + a_{nj}v_n + a_{(n+1)j} \\ &= \sum_{i=1}^{n} a_{ij}v_i + a_{(n+1)j} \end{aligned} \tag{5}$$

The linguistic equivalent of a rule is formulated accordingly:

$$\text{IF } F_{1j}(v_1) \text{ AND } F_{2j}(v_2) \text{ AND .. AND } F_{nj}(v_n) \text{ THEN } f_j(\overrightarrow{\mathbf{v}}_t) \tag{6}$$

The membership functions of the rule are non-linear Gaussian functions, with μ_{ij} the mean value and σ_{ij}^2 the variance for the membership function F_{ij}, which are calculated accordingly:

$$F_{ij}(v_i) := e^{-\frac{(v_i-\mu_{ij})^2}{(2\sigma_{ij}^2)}} \tag{7}$$

The antecedent part of the rule j determines the weight w_j accordingly:

$$w_j(\overrightarrow{\mathbf{v}}_t) := \prod_{i=1}^{n} F_{ij}(v_i) \tag{8}$$

The projection from input $\overrightarrow{\mathbf{v}}_t := (v_1, v_2, .., v_n)$ onto the classifiable one-dimensional set is a weighted sum average, which is a combination of fuzzy inference and defuzzification. The weighted sum average is calculated according to the rules $j = 1, .., m$ as follows:

$$\mathbf{S}(\overrightarrow{\mathbf{v}}_t) := \frac{\sum_{j=1}^{m} w_j(\overrightarrow{\mathbf{v}}_t) f_j(\overrightarrow{\mathbf{v}}_t)}{\sum_{j=1}^{m} w_j(\overrightarrow{\mathbf{v}}_t)} \tag{9}$$

FIS with Covariant MFs In ubiquitous computing we are mostly dealing with highly correlated data, especially when multi axes accelerometers are used. With multivariate membership functions (MF) the data cannot be covered sufficiently, therefore more functions are needed (fig. 9, left). More membership functions lead to more rules, which leads to a increased calculation effort. Since multivariate MF cannot adapt to the shape of covariant data, the data is not accurately represented by the MF, and therefore a separation of different classes results in a bigger error [31]. On the other hand, multidimensional covariant MFs are not as intuitively interpretable as separate multivariate MFs. In this thesis only an interpretation of the model on rule level is needed, which is still possible with covariate MFs.

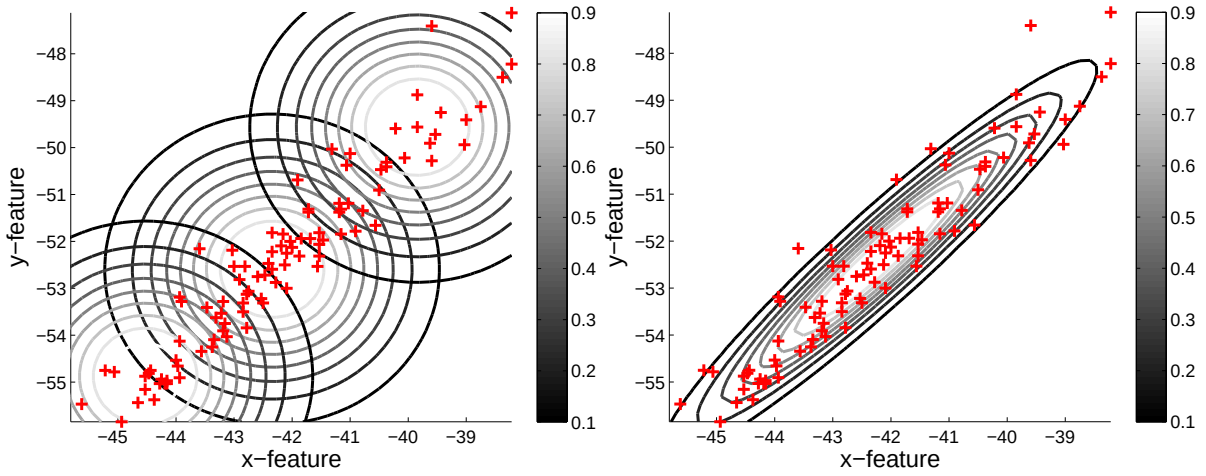


Figure 9: Contour plot of multivariate (left) and covariate (right) Gaussian membership functions.

This reasons argue strongly for using covariant MFs (fig. 9, right), which result in less rules and smaller error. The covariant MFs used here are defined accordingly:

$$\mu_j(\overrightarrow{\mathbf{v}}_t) := e^{-\frac{1}{2}(\overrightarrow{\mathbf{v}}_t - \overrightarrow{\mathbf{m}}_j)\Sigma_j^{-1}(\overrightarrow{\mathbf{v}}_t - \overrightarrow{\mathbf{m}}_j)^T} \tag{10}$$

With covariant MFs the whole input of each rule is handled by one MF, which results in a simplified linguistic rule:

$$\text{IF } \mu_j(\overrightarrow{\mathbf{v}}_t) \text{ THEN } f_j(\overrightarrow{\mathbf{v}}_t) \tag{11}$$

The whole antecedent part of each rule was multiplied in the usual TSK-FIS (eqn. 8) inference to get the respective weight, but with the covariant MF's the function is already the weight. The resulting formula for the covariant TSK-FIS is defined, as follows:

$$\mathbf{S}(\overrightarrow{\mathbf{v}}_t) := \frac{\sum_{j=1}^{m} \mu_j(\overrightarrow{\mathbf{v}}_t) f_j(\overrightarrow{\mathbf{v}}_t)}{\sum_{j=1}^{m} \mu_j(\overrightarrow{\mathbf{v}}_t)} \tag{12}$$

Recurrence in FIS Mapping The outcome of the mapping at time t is fed back as additional input dimension for the TSK-FIS mapping at $t+1$. This results in a changed system equation, which is defined in the following:

$$\mathbf{S}\left(\overrightarrow{\mathbf{v}}_{t+1}, \mathbf{S}(\overrightarrow{\mathbf{v}}_t)\right) := \frac{\sum_{j=1}^{m} \mu_j(\overrightarrow{\mathbf{v}}_{t+1}) f_j(\overrightarrow{\mathbf{v}}_{t+1}) + \mu_{n+1}(\mathbf{S}(\overrightarrow{\mathbf{v}}_t)) f_{n+1}(\mathbf{S}(\overrightarrow{\mathbf{v}}_t))}{\sum_{j=1}^{m} \mu_j(\overrightarrow{\mathbf{v}}_t) + \mu_{n+1}(\mathbf{S}(\overrightarrow{\mathbf{v}}_t))} \tag{13}$$

The recurrence not only delivers a reliability measure, but also stabilizes and improves the mapping accuracy [23]. The output of the TSK-FIS is assigned to a tuple of class and fuzziness, which represents the reliability of the classification. Instead of 'Recurrent TSK-FIS' the simpler term RFIS is used in the remainder of this thesis.

The output of the RFIS mapping needs to be assigned to a class identifier. This is done in a separate classification process, where also a reliability measure is calculated. The classification function is defined in the following.

3.1.4 Fuzzy Classification

The processing queue up to this point provides a dimensionality reduction and a time dependent compression of the sensor measurement stream. After the mapping function provides an output with only one dimension, a classification process can take over.

Class Tuples A class c_i is defined as a tuple of a numerical class identifier i and three char characters 'XYZ' uniquely identifying the activity. Each char triple has natural language semantics, which are provided in semantics tables and described along with examples later in this thesis. The first char partly identifies the **conditionality** of the mobile phone or the user e.g. 'P' stands for the user having the phone in her 'pocket' or 'H' in her 'hand'. The latter two characters mostly describe the activity, e.g. 'ST' stands for 'standing' or 'HO' for 'holding'. In this manner the set $\mathcal{C}$ of activity classes is defined:

$$\mathcal{C} := \left\{c_i \in \mathbb{Z}^+ \times \mathbb{S}^3 | c_i = (i, \text{'XYZ'}); i = 1, .., o; \text{'XYZ'} \in \mathbb{S}^3\right\} \tag{14}$$

Here $\mathbb{S}$ is the set of all possible characters, which in this case only includes capital alphabetic letters or positive integer numerical values. The amount of numerical class identifiers is limited to o, but this is just an arbitrary number, which can be set to a high value or be changed on demand. This is necessary, since the proposed novel modular activity recognition architecture is extensible and therefore the number of activity classes is not known at design time. More on the extensibility is described in subsection 4.2.

Fuzzy Numbers - Motivation The output of the RFIS numerically encodes the class identifiers $i = 1, .., o$. The assignment of the RFIS mapping result to a numerical class identifier i is done fuzzily, so the outcome is not only a class identifier, but also a membership, representing the reliability of the classification process. According to the numerical class identifier i the respective class tuple $c_i = (i, \text{'XYZ'})$ can be chosen. Each numerical class identifier i is interpreted by a triangle-shaped fuzzy number $\tilde{i}$ with its mean is the identifier i itself - see fig. 10.

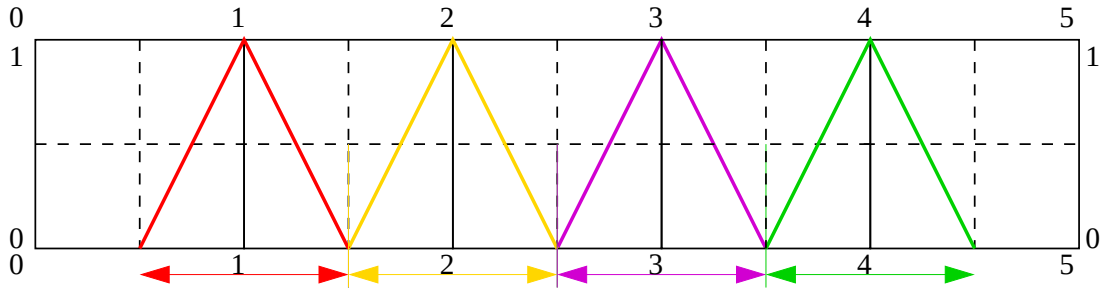


Figure 10: Fuzzy numbers identifying class membership and fuzzy uncertainty level

Fuzzy Numbers - Theory A fuzzy number is a function which meets the following constraints in definition 3.

Definition 3 (*Fuzzy Number*)

A fuzzy umber is a fuzzy quantity $\tilde{A}$ with

1. *$\mu_{\tilde{A}}(x) = 1$ for exactly one $x \in \mathbb{R}$*
2. *$\tilde{A}$ is convex.*

There is a special LR-Description of fuzzy numbers, which is used in the following and defined in definition 4.

Definition 4 (*LR-Description of Fuzzy Numbers*)

A fuzzy number $\tilde{A}$ has a LR-description if there are two functions L and R and scalars $\alpha > 0$, $\beta > 0$ and m, so that

$$\mu_{\tilde{A}}(x) = \begin{cases} L\left(\frac{m-x}{\alpha}\right) & \text{, when } x < m \\ 1 & \text{, when } x = m \\ R\left(\frac{x-i}{\beta}\right) & \text{, when } x > m. \end{cases}$$

m is the mean, α and β the left and right spread of $\tilde{A}$

Here a special triangular shaped fuzzy number is used, which is shown in figure 10 as example and defined in definition 4 in combination with equation 15.

$$L(x) = R(x) = \begin{cases} (1-x) & \text{for } 0 < x < 1 \\ 0 & \text{else.} \end{cases} \tag{15}$$

More on the topic of fuzzy numbers can be found in [8].

Fuzzy Numbers - Praxis Since positive integer numbers are used for the numerical class identifiers i, where the numbering starts with one and ends with o without any gaps, α and β are defined as $\alpha = \beta = \frac{1}{2}$. This results in symmetrical triangular fuzzy numbers with no overlapping in-between the numbers, which is described through the following equation:

$$\mu_{\tilde{i}}(x) = \begin{cases} \max\left(0, 1 - 2(i-x)\right) & \text{, when } x < i \\ 1 & \text{, when } x = i \\ \max\left(0, 1 - 2(x-i)\right) & \text{, when } (i + \frac{1}{2}) > x > i \\ \lim\limits_{k \to \infty} \frac{1}{k} & \text{, when } x = (i + \frac{1}{2}). \end{cases} \tag{16}$$

If there would be overlappings between the MFs of the fuzzy numbers $\tilde{i}$, some classifications would never have a reliability measure in the full range of the interval $[0, 1]$. Also, for the right side of the triangular membership function of the fuzzy number $\tilde{i}$, there was a line added to the equation 15. This is because of the border exactly in-between the fuzzy number where the following would be valid instead:

$$\begin{array}{ccc} \mu_i(x) & = & \mu_{i+1}(x) \quad \text{, with } i < o \\ \max\left(0, 1 - 2(x-i)\right) & = & \max\left(0, 1 - 2((i+1) - x)\right) \quad \text{, with } x = i + \frac{1}{2} \\ 0 & = & 0 \end{array}$$

In this case of $x = i + \frac{1}{2}$ the usual definition of the triangular shaped fuzzy number $\tilde{i}$ would lead to an inability to decide during the classification process. So, the last case in the equation 15 for $x = i + \frac{1}{2}$ determines that the membership function $\mu_{\tilde{i}}(x)$ would have an output which is infinitesimally close to zero, but is not zero. In case of a finite machine as the CPU of a mobile phone, this number is limited through the used arithmetic.

Classification Process Now the classification process has to be defined, which chooses the best fitting triangular fuzzy number $\tilde{i}$, decides on the respective class tuple $c_i = (i, \text{'XYZ'})$ and calculates the reliability of the classification $\mu_{\tilde{i}}(x)$. The classification process is defined as a function $K : \mathbb{R} \to \mathcal{C} \times [1, 0]$ in the following:

$$\mathbf{K}(x) = \begin{cases} (c_1, \mu_{\tilde{1}}) & \text{, when } \mu_{\tilde{1}}(x) = \max_{\tilde{i}} \left(\mu_{\tilde{i}}(x)\right) \\ \vdots & \vdots \\ (c_o, \mu_{\tilde{o}}) & \text{, when } \mu_{\tilde{o}}(x) = \max_{\tilde{i}} \left(\mu_{\tilde{i}}(x)\right). \end{cases} \tag{17}$$

After the classification the output is a tuple of class c_i and membership $\mu_{\tilde{i}}$ to this class, where the membership represents the reliability of the classification process. The class $c_i = (i, \text{'XYZ'})$ again is a tuple of numerical class identifier i and char triple 'XYZ' distinctively identifying the activity.

Classification Module In summary the sensor processing queue and therefore the sensor data classification is a combination of functions partly repeatedly described in the following equations:

$$\mathbf{M}(\overrightarrow{\mathbf{v}}_t) = \mathbf{K}(\mathbf{S}(\overrightarrow{\mathbf{v}}_t, \mathbf{S}(\overrightarrow{\mathbf{v}}_{t-1}))) \mapsto (c_i, \mu_{\tilde{i}}) \tag{18}$$

$$\overrightarrow{\mathbf{v}}_t = (v_{1,t}, v_{2,t}, .., v_{n,t})$$

In this thesis the focus lies on the mapping function and the classification process and not the feature extraction methods. Evaluating all parts with all their degrees of freedom together would go beyond the scope of this thesis. The mapping function and the classification process are therefore the variable parts of the processing queue and are bound together as a module $\mathbf{M}$. What a module is in particular and how modules can be variable is described in subsection 3.2.

Since the classification process not only provides a class output, but also a reliability measure describing the confidence of the classification, a filtering on the reliability can improve the classification accuracy in average. How the filtering works is described in the following.

3.1.5 Reliability Filter

The reliability measure provides an unique advantage of improving the activity recognition accuracy on average. Here a filtering upon the reliability measure can separate the reliable from the unreliable classifications, thus some of the incorrect classifications can be sorted out. Of course, the reliability measure is not absolute, so there always will be false positives and false negatives. But on average the classification accuracy can be improved. Another issue is, that the amount of classifications at the total end of the processing queue is lowered depending on the filter threshold. Also, the classification output after the filtering is not continuous any more, so there will be smaller or bigger gaps in-between. The reliability measure, how it can be derived and what is its information content is discussed and experimentally evaluated in subsection 5.3.

Filter Function The filter is defined as function $\mathbf{F} : \mathcal{C} \times [1, 0] \to \mathcal{C} \times [1, 0] \cup \emptyset$ depending on the threshold τ in the following:

$$\mathbf{F}^{\tau}(c_i, \mu_{\tilde{i}}) = \begin{cases} (c_i, \mu_{\tilde{i}}) & \text{, when } \mu_{\tilde{i}} \geq \tau \\ \emptyset & \text{, else.} \end{cases} \tag{19}$$

If the reliability measure $\mu_{\tilde{i}}$ is beneath threshold τ the function $\mathbf{F}$ has the empty set $\emptyset$ as output, which is in a technical system no output at all. The filter can be switched off if the threshold τ is set to zero.

Receiver Operator Characteristic (ROC) In order to determine the threshold value τ for the filtering, the method known as "Receiver Operator Characteristic (ROC)" [54] can be used. ROC graphs are generally useful for organizing classifiers and visualizing their performance. Here a two-class problem, whether the classification was correct or incorrect, leads to the determination of a threshold value τ. According to this threshold τ a confusion matrix can be set up, where the true positives (TP), true negatives (TN), false negatives (FN) and false positives (FP) are listed. Here, the following distinctions are made:

True Positive (TP): The reliability value is above the threshold τ and the classification was indeed correct.

True Negative (TN): The reliability measure is below the threshold τ and the classification was incorrect.

False Positive (FP): The reliability measure is above the threshold τ, but the classification was incorrect.

False Negative (FN): The reliability measure is below the threshold τ, but the classification was correct.

To determine these probabilities and the threshold τ, probability distributions over the correct and incorrect classifications of a test set are placed separately. As probability distribution Gaussian density functions are used. The normal distributions are identified through a maximum likelihood estimation. The results for an exemplary small test set (fig. 11) of such an estimation can be seen in figure 12.

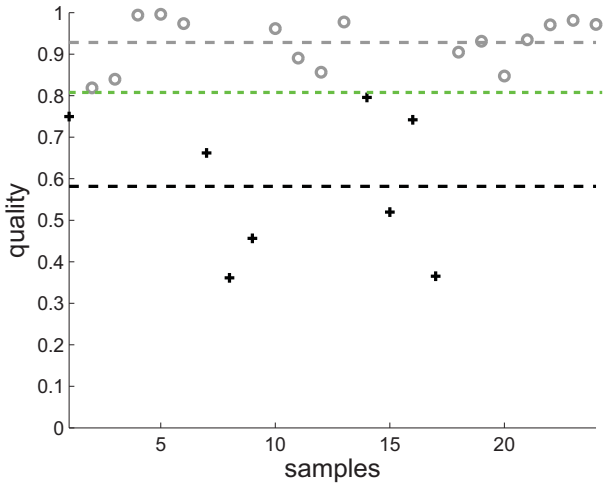


Figure 11: Example of reliability measures for a test set of 24 classifications for correct (o) and incorrect (+) activity recognitions and mean values (dashed lines).

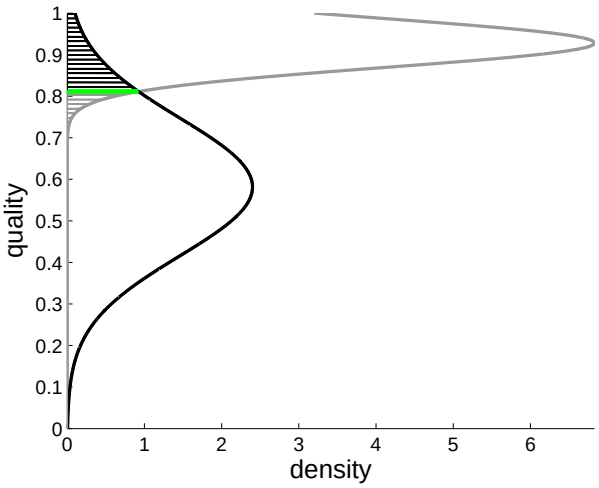


Figure 12: Gaussian density functions for correct (grey) and incorrect (black) classified data with marked threshold value (green) and hatched median cuts.

Determining Threshold Value The threshold value is now placed in the intersection of the two normal distributions of correct and incorrect classified data, which is described through the following equation:

$$
\begin{aligned}
p_r(\tau_I) &= p_w(\tau_I) \\
\frac{1}{2\sigma_r\sqrt{2\pi}} e^{-\frac{(\tau_I-\mu_r)}{(2\sigma_r^2)}} &= \frac{1}{2\sigma_w\sqrt{2\pi}} e^{-\frac{(\tau_I-\mu_w)}{(2\sigma_w^2)}} & |\cdot \tfrac{2\sigma_w\sqrt{2\pi}}{1} \\
\frac{\sigma_w}{\sigma_r} e^{-\frac{(\tau_I-\mu_r)}{(2\sigma_r^2)}} &= e^{-\frac{(\tau_I-\mu_w)}{(2\sigma_w^2)}} & |\ln \\
\ln\frac{\sigma_w}{\sigma_r} - \frac{(\tau_I-\mu_r)}{(2\sigma_r^2)} &= -\frac{(\tau_I-\mu_w)}{(2\sigma_w^2)} & |+\frac{(\tau_I-\mu_r)}{(2\sigma_r^2)} \\
\ln\frac{\sigma_w}{\sigma_r} &= \frac{(\tau_I-\mu_r)}{(2\sigma_r^2)} - \frac{(\tau_I-\mu_w)}{(2\sigma_w^2)} \\
\ln\frac{\sigma_w}{\sigma_r} &= \frac{\sigma_w^2(\tau_I-\mu_r)-\sigma_r^2(\tau_I-\mu_w)}{2\sigma_r^2\sigma_w^2} \\
\ln\frac{\sigma_w}{\sigma_r} &= \frac{\tau_I(\sigma_w^2-\sigma_r^2)-\sigma_w^2\mu_r+\sigma_r^2\mu_w}{2\sigma_r^2\sigma_w^2} & |+\frac{\sigma_w^2\mu_r-\sigma_r^2\mu_w}{2\sigma_r^2\sigma_w^2} \\
\ln\frac{\sigma_w}{\sigma_r} + \frac{\sigma_w^2\mu_r-\sigma_r^2\mu_w}{2\sigma_r^2\sigma_w^2} &= \frac{\tau_I(\sigma_w^2-\sigma_r^2)}{2\sigma_r^2\sigma_w^2} & |\cdot\frac{2\sigma_r^2\sigma_w^2}{(\sigma_w^2-\sigma_r^2)} \\
\frac{2\sigma_r^2\sigma_w^2\ln\frac{\sigma_w}{\sigma_r}+\sigma_w^2\mu_r-\sigma_r^2\mu_w}{(\sigma_w^2-\sigma_r^2)} &= \tau_I
\end{aligned}
\tag{20}
$$

In 20 μ_r is the mean and σ_r^2 is the variance for the density function $p_r(x)$ of correct classified data pairs. Accordingly, μ_w and σ_w^2 for $p_w(x)$ of incorrect classified data pairs. The threshold of the intersection is τ_I (see example in fig. 12), which is the best trade-off between FP and FN (see also fig. 11). If the developer decides that the classifier's precision is more important than recall, the threshold τ can be set to a higher value. This also increases the FN rate and therefore the amount of the overall passing classifications. The same applies for the FN rate when the threshold τ is set to a lower value than τ_I.

The filtering can improve the recognition rates, but also reduces the amount of classifications, which are possibly handed over to the next processing step or application. This is a trade-off which can be influenced through the threshold. The whole Processing queue is expressed through the following equation:

$$
\begin{aligned}
\mathbf{F}^\tau\left(\mathbf{M}(\overrightarrow{\mathbf{v}}_t)\right) &= \mathbf{F}^\tau\left(\mathbf{K}(\mathbf{S}(\overrightarrow{\mathbf{v}}_t, \mathbf{S}(\overrightarrow{\mathbf{v}}_{t-1})))\right) \\
&= \mathbf{F}^\tau\left(\mathbf{K}(\mathbf{S}(v_{1,t}, v_{2,t}, .., v_{n,t}, \mathbf{S}(\overrightarrow{\mathbf{v}}_{t-1})))\right) \\
&= \mathbf{F}^\tau\left(\mathbf{K}(\mathbf{S}(m(\overrightarrow{x}_{1,t}), m(\overrightarrow{y}_{1,t}), m(\overrightarrow{z}_{1,t}), \sigma^2(\overrightarrow{x}_{1,t}), \sigma^2(\overrightarrow{y}_{1,t}), \sigma^2(\overrightarrow{z}_{1,t}), \mathbf{S}(\overrightarrow{\mathbf{v}}_{t-1})))\right)
\end{aligned}
$$

In addition to the explained processing queue, a bit-vector masking of the mapping function can be done. This step has only indirect influence on the processing queue, which is why it is explained last in this subsection.

3.1.6 Bit-Vector Masking

When performing activity recognition on mobile phones, there are certain changes that require a reaction. E.g. user, clothes, seasons, behavior, physical and mental conditions change, which leads to different sensor patterns and therefore demand an adaption of the mapping function. More on this topic is discussed and evaluated in section 4.1. Since these changes are mostly non permanent or just need a marginal variation of the mapping function, a modification method is needed, which is only temporal and preserves the original mapping capabilities. Therefore, a novel bit-vector masking (see fig. 13) is introduced, which just switches the dimensions of each rule of the RFIS mapping functions on or off. A representation of the RFIS is needed that specifies the dimensionality of each rule, which is in the current system either active or inactive. The two states to be specified allow the representation to be a bit vector, which is due to its efficient memory usage especially easy to store and process on platforms with limited **resources**. A genetic algorithm is used to identify the bit-vector according to changed conditions. The bit-vector identification is described in subsection 3.3.7.

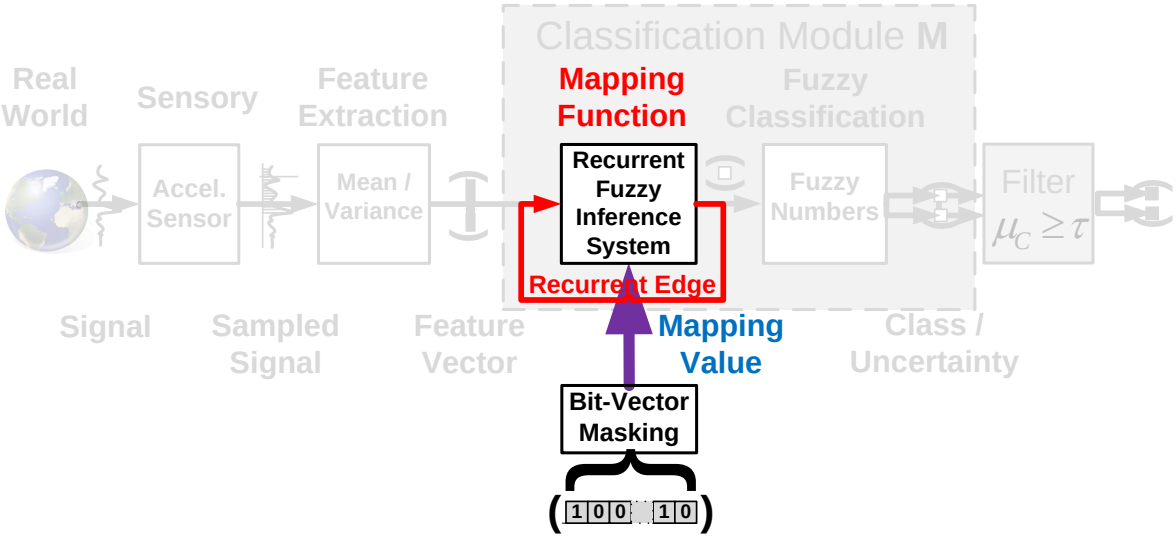


Figure 13: Classification system architecture with **additional** bit-vector masking.

Interpretation Function First, a function is defined that maps the general RFIS $\mathbf{S}$ according to the bit specification onto a modified RFIS $\mathbf{S}'$ with modified rule dimensionality. The mapping function $\mathcal{I}$ is rather an interpretation, than a modification of the FIS, since it only temporarily *switches* the inputs of the rules on or off, and does not modify the topology of the FIS permanently. The interpretation function $\mathcal{I}$ is defined, as follows:

$$\mathcal{I} : \begin{cases} \mathbb{F}_{RFIS}(\mathbb{R}^{n+1}, \mathbb{R}) \times \{0,1\}^{m(n+1)} & \longrightarrow & \mathbb{F}_{RFIS}(\mathbb{R}^{n+1}, \mathbb{R}) \\ (\mathbf{S}(\overrightarrow{\mathbf{v}}_t, \mathbf{S}(\overrightarrow{\mathbf{v}}_{t-1})), bit_{\mathbf{S}}) & \longmapsto & \mathbf{S}'(\overrightarrow{\mathbf{v}}_t, \mathbf{S}'(\overrightarrow{\mathbf{v}}_{t-1})) \end{cases} \tag{21}$$

The interpretation $\mathcal{I}$ maps from the space of RFIS functions $\mathbb{F}_{RFIS}$ onto the same space, where $\mathbb{F}_{RFIS}(\mathbb{R}^{n+1}, \mathbb{R})$ is a set of FIS functions mapping from $\mathbb{R}^{n+1}$ onto $\mathbb{R}$.

Bit-Vector Specification The bit vector specifies the inputs of the rules, which is described through the following example:

$$\begin{array}{ccc} & \overbrace{\text{rule } 1} & \dots & \overbrace{\text{rule } m} \\ bit_{\mathbf{S}} = & (1,0,1,..,0,1,1, & \dots & ,0,1,0,..,1,0,0) \\ & (v_1,-,v_3,..,-,v_n,\mathbf{S}'(\overrightarrow{\mathbf{v}}_{t-1})) & \dots & (-,v_2,-,..,v_{n-1},-,-) \\ & \mu'_1(v_1,v_3,..,v_n,\mathbf{S}'(\overrightarrow{\mathbf{v}}_{t-1})) & \dots & \mu'_m(v_2,..,v_{n-1}) \end{array} \tag{22}$$

Only the antecedent membership functions are masked since they specify the weight of each dimension in each rule. If the linear functional consequence would be masked the output of the RFIS would be changed in an unpredictable way. Nevertheless, the bit vector masking for the MFs and the linear functional consequence are analyzed in [31] and in subsection 4.1. Since all dimensions are processed in one covariant Gaussian function, the masking directly changes the dimensions in the covariance matrix. The i'th row and i'th column of the inverted covariance matrix Σ_j^{-1} of membership function μ_j for rule j are set to zero, if the i'th dimension of this rule is switched off. The following example should illustrate the procedure:

$$\Sigma_j^{-1} = \begin{pmatrix} \sigma_{1,1}^{-1} & \dots & \sigma_{1,i}^{-1} & \dots & \sigma_{1,n+1}^{-1} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \sigma_{i,1}^{-1} & \dots & \sigma_{2,i}^{-1} & \dots & \sigma_{2,n+1}^{-1} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \sigma_{n+1,1}^{-1} & \dots & \sigma_{n+1,i}^{-1} & \dots & \sigma_{n+1,n+1}^{-1} \end{pmatrix} \longmapsto \Sigma_j^{'-1} = \begin{pmatrix} \sigma_{1,1}^{-1} & \dots & 0 & \dots & \sigma_{1,n+1}^{-1} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \sigma_{n+1,1}^{-1} & \dots & 0 & \dots & \sigma_{n+1,n+1}^{-1} \end{pmatrix} \tag{23}$$

There are two ways to optimize the bit-vector upon: one is the optimization according to the mean squared error (MSE) and the other one is the percentage of correct classifications. Both are calculated on a data set, which represents the changes of users, clothes, seasons, behaviors, physical and mental conditions. Details about the identification method are presented in subsection 3.3.7 and the approach is evaluated in detail throughout section 5.

3.2 Modular Classification

The key contribution in this thesis to the research field of activity recognition on mobile phones is the modularity of the activity recognition process. Especially in the research fields of cognitive science the utilization of modular classifiers and pattern recognizers is very well researched. Some of the contributions are listed and described in subsection 2.3, along with an outline of the differences to the proposed approach in this thesis. The evaluation of the activity recognition architecture along with the challenges of activity recognition on mobile phones follows in section 5.

First, the classification module is defined in conjunction with the method for a module transition. Here the novel concept of a *complementary class* is introduced, which on recognition is triggering a module transition.

Second, a dynamic queue of classification modules is described and how it is used in the activity recognition process. This concept is the most simple in order to have the classifier modules schedule themselves. The dynamic queue concept is used throughout this thesis as main scheduling method.

Third, a probabilistic possibility for the module scheduling is briefly discussed. Since this method requires an additional model to be identified and on occasion to be extended or modified, this approach is introduced without any following evaluation.

Lastly, other techniques are lined out which could further improve the dynamic queue concept, the probabilistic approach or can be used instead. These methods also will not be investigated further in this thesis.

3.2.1 Classification Module

In subsection 3.1 the data processing queue was described. Two constants have been determined, the used sensor and the feature extraction methods. The variables in the process are the mapping function and the fuzzy classification, since these depend primarily on what activity classes need to be recognized and secondly on the various boundary conditions influencing the activity recognition. These two chains in the processing queue that need to be variable are bound together and are defined as a classification module (fig. 14).

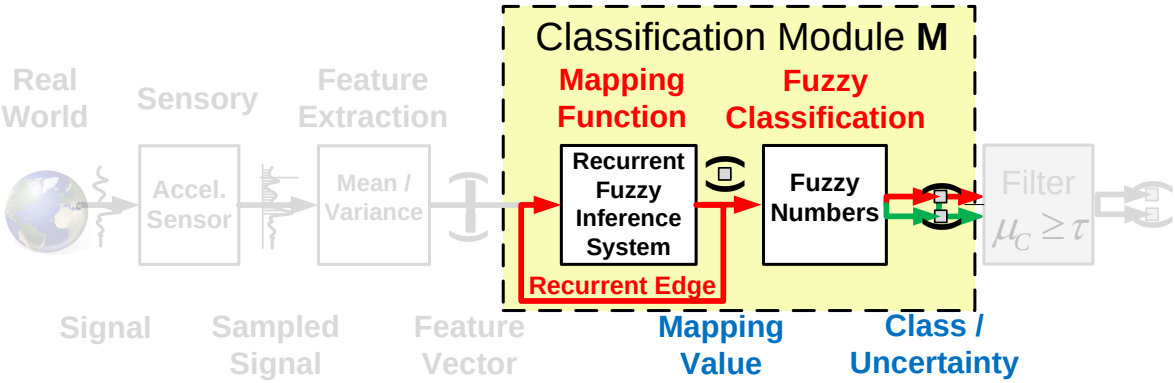


Figure 14: Illustration of the module concept and its part of the processing queue.

Classes Subsets The activities which should be recognized on the mobile phone are defined as a set $\mathcal{C}$ of class tuples $c_i = (i, \text{'XYZ'})$ in 3.1.4. Instead of using one classifier module to classify on all classes $\mathcal{C}$, several modules $\mathbf{M}_k : \mathbb{R}^n \rightarrow \mathcal{C}_k$ (with $k = 1, .., p$) are used, each classifying on a small subset $\mathcal{C}_k \subseteq \mathcal{C}$ of classes. The subsets $\mathcal{C}_k$ can be chosen according to the classes $c_{kl} \in \mathcal{C}_k$ semantics, therefore each subset $\mathcal{C}_k$ could have its own meta semantic. This meta semantic is called *conditional context*, which is further described and motivated in subsection 4.5.

The subsets $\mathcal{C}_k$ need to be defined, since the numerical class identifiers $i \in \mathbb{Z}^+$ can not be mapped one-to-one. The later described *complementary class*, which is needed to recognize inter module transitions, is identified in every module with 0. Due to the nature of the mapping and the reliability measure, all classes and therefore fuzzy

numbers need to cover the same input area. If one class would have a bigger input space than others, it would be preferred, since a larger mapping error is covered. Therefore each subset's $\mathcal{C}_k$ first numerical class identifier needs to be $i = 1$. The subsets $\mathcal{C}_k$ are defined in the following:

$$\mathcal{C}_k := \left\{ c_{kl} \in \mathbb{Z}^+ \times \mathbb{S}^3 | c_{kl} = (k, l, \text{'XYZ'}); l = 1, .., o_k; i \hat{=} \sum_{j=1}^{k} o_j + l; \text{'XYZ'} \in \mathbb{S}^3 \right\} \tag{24}$$

The classes c_{kl} are now represented through a triple of module identifier k, numerical class identifier l and the usual char triple 'XYZ' uniquely identifying the activity. The translation of the new class triples c_{kl} into the original class tuples c_i is done according to tables (see examples in section 4) and the char triple 'XYZ'. This offers the most flexibility for an extensible modular classifier, where for each module it can be decided which classes it should recognize. The classes $c_{kl} = (k, l, \text{'XYZ'})$ are uniquely identified through the char triples 'XYZ', which remain the same as with the old class set definition $\mathcal{C}$.

Complementary Class In order to not only recognize the respective classes $c_{kl} \in \mathcal{C}_k$, but also the transition between classifiers $\mathbf{M}_k$, each module yields a *complementary class* $\overline{c}_k$ as well. The complementary class $\overline{c}_k$ indicates the inability of the module $\mathbf{M}_k$ to classify the data onto a native class $c_{kl} \in \mathcal{C}_k$. The class $\overline{c}_k$ would therefore be the complement of all classes $c_{kl} \in \mathcal{C}_k$, but in practice the complement is fuzzy.

Since the *complementary class* is also identified through machine learning on data, which should be classified through other modules M_j with $j \neq k$, it does not cover the whole complementary input space. An alpha cut of the MFs of the RFIS mapping function $\mathbf{S}_k$ of module $\mathbf{M}_k$ would deliver a crisp decision for the feature vector $\mathbf{v}_t$ if it belongs to the classes of $\mathcal{C}_k$ or not. Therefore, a crisp *complementary class* could be defined, which is exemplary visualized in figure 15 on the left. The problem here are the MFs of the RFISs $\mathbf{S}_j$ of the other module $\mathbf{M}_j$ with $j \neq k$, which sometimes overlap with the MFs of another modules mapping function $\mathbf{S}_k$. This leads to an α in the alpha cuts, which needs to be variable for each of the different overlapping areas sizes. If the α is set to a low value, the decision of which module the input feature vector belongs to is sometimes not determinable - see example in figure 15 on the right side. On the other hand if the α is higher than all of the MFs intersections, the cuts would disqualify a lot of feature vectors from being classified by any of the modules. Also, a determination of

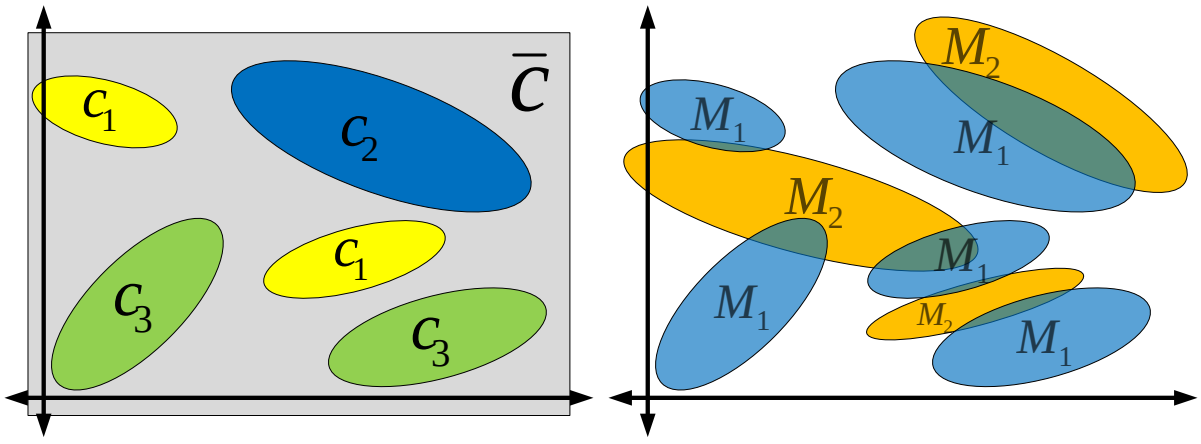


Figure 15: Examples for alpha cuts: alpha cuts of different MFs for different classes c_i of one module and resulting complementary class $\overline{c}$ (left); alpha cuts of MFs of two different modules $\mathbf{M}_1$ and $\mathbf{M}_2$ with overlappings (right).

a complementary class on an alpha cut is only possible with fuzzy set based mapping functions and could e.g. not be applied to a neural network. This thesis focuses on the general applicability of the modularity and other novel features presented here, so this approach would be disqualified due to inflexibility.

The complementary class is defined as $\overline{c}_k = (k, 0, '')$, since it has no semantics and just indicates a module switch. Also, the class identifier 0 was chosen because the number was not used before, is nearest predecessor of

the first class identifier $l = 1$ and still enables the usage of nonnegative integer numbers $\mathbb{Z}^*$ as numerical class identifiers. The second circumstance is mandatory as previously explained and the third is of profit for an efficient implementation of the activity recognition. How each mapping function $\mathbf{S}_k$ is trained for the complementary class is explained in subsection 3.3. The module transition through the recognition of the complementary class is described in subsection 3.2.2.

Modular Classification According to the changes in the set of classes $\mathcal{C}_k$ and the newly used *complementary class* $\overline{c}_k$ the classification function $\mathbf{K}$ (eqn. 26) has to be altered. But first the membership functions $\mu_{\tilde{i}}(x)$ (eqn. 16) index needs to be changed from the previously used identifier i to the now module based class identifier l, which leads to the following new membership function:

$$\mu_{\tilde{l}}(x) = \begin{cases} \max\left(0, 1 - 2(l - x)\right) & \text{, when } x < l \\ 1 & \text{, when } x = l \\ \max\left(0, 1 - 2(x - l)\right) & \text{, when } (l + \frac{1}{2}) > x > l \\ \lim\limits_{p \to \infty} \frac{1}{p} & \text{, when } x = (l + \frac{1}{2}). \end{cases} \tag{25}$$

Now the classification function $\mathbf{K} : \mathbb{R} \to \mathcal{C}$ is changed to the module based notation $\mathbf{K}_k : \mathbb{R} \to \mathcal{C}_k \cup \overline{c}_k$ with the additional classification onto the complementary class $\overline{c}_k$ accordingly:

$$\mathbf{K}_k(x) = \begin{cases} (c_{k1}, \mu_{\tilde{1}}) & \text{, when } \mu_{\tilde{1}}(x) = \max\left(\max\limits_{\tilde{l}}\left(\mu_{\tilde{l}}(x)\right), \mu_{\tilde{0}}(x)\right) \\ \vdots & \vdots \\ (c_{ko_k}, \mu_{\tilde{o}_k}) & \text{, when } \mu_{\tilde{o}_k}(x) = \max\left(\max\limits_{\tilde{l}}\left(\mu_{\tilde{l}}(x)\right), \mu_{\tilde{0}}(x)\right) \\ (\overline{c}_k, \mu_{\tilde{0}}) & \text{, when } \mu_{\tilde{0}}(x) = \max\left(\max\limits_{\tilde{l}}\left(\mu_{\tilde{l}}(x)\right), \mu_{\tilde{0}}(x)\right). \end{cases} \tag{26}$$

The classification modules $\mathbf{M}_k$ on the set $\mathcal{C}_k$ are now defined. Also, the complementary classes $\overline{c}_k$ purpose was defined and explained. The response on the complementary class recognition needs still to be specified.

3.2.2 Dynamic Queue of Classification Modules

There are many ways of organizing the modular classifiers, each has advantages and disadvantages. The one which is favored, applied and evaluated throughout this thesis is the dynamic queue approach.

All classifiers are chained in a dynamic queue, where the last classifier classifying a class different from the *complementary* one is put first in queue. The idea behind this re-organization of the classifier queue is, that modules are successive in the queue, which are successive in recognition of input features. Therefore, when the currently *active* module recognizes the complementary class $\overline{c}_k$, preferably not the whole queue needs to be tested for capabilities of classifying this feature vector $\overrightarrow{v}_t$, but only the next.

Each of the activity classification modules not only recognizes the respective classes, but also a so called *complementary class* $\overline{c}_k$. If this *complementary class* is recognized, another module becomes activated. In this manner the *complementary class* indicates whether a classification module can or cannot classify the feature vector input. All classification modules are organized in a dynamic queue, where the last module classifying on a class different from the complementary one gets sorted in front of the queue. Only the first module in the queue is active and therefore only a fraction of the overall activity recognizer needs to be calculated. This reduces calculation effort and therefore power consumption. For better understanding of the dynamic queue of classification modules, an example is displayed in figure 16 and described in the following:

1. The classification modules queue starts at time $t = 0$ (before classification of first feature vector) where the modules are sorted according to their module number. Only the first module in the queue is active and gets to classify the incoming feature vectors.

2. At time $t = X$ the incoming feature vector gets classified by the first module onto the *complementary class*. This indicates, that this module can not classify this feature vector onto a native class.

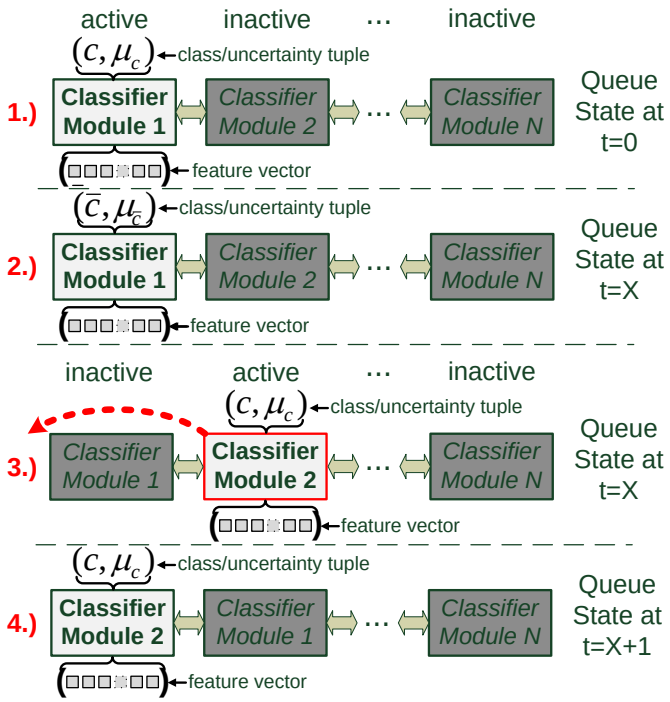


Figure 16: State of modular classifiers in queue at time $t = 0$, $t = X$ and $t = X + 1$

3. The next module in the queue then becomes activated and tries to classify the same feature vector. In this example it succeeds and classifies the feature vector on a class different from the complementary one. This module number two now gets put first in queue.

4. In the next time step $t = X + 1$ (new feature vector) the first classifier module is module number two. This module remains first in queue until it classified onto the *complementary class*.

Due to mis-classifications or feature vectors of classes not being recognized by any of the modules it is possible, that every module in the queue classifies onto the *complementary class*. In this case, the overall output of the queue is the *complementary class* and the order of the modules in the queue remains the same as before.

3.2.3 Probability of Module Transitions

Another method to organize and schedule the modules is based on conditional probabilities. The first ideas about this approach had been published in [29]. The underlying modular classification with fuzziness is modeled in this approach according to conditional probabilities. All modules are connected to each other (fig. 17), where the probability of a transition from one module to another is individual. Therefore, if a module is recognizing the *complementary class* not a next module in a dynamic queue is activated, but the one with the highest probability of being the successor. The idea about this approach is, that some transitions have high probability and others are less likely. E.g. if currently the module for the *conditional context phone is lying on table* is active, and a state transition is occurring due to the recognition of the *complementary class*, a module with the meta semantic *user is holding the phone* is more likely than *phone in pants pocket*. This is due to the fact, that the user first has to pick up the phone and only then can put it into a pocket.

Transition Probability Matrix If the successor that is activated after the currently active module is recognizing the *complementary class*, it is not capable of classifying the feature vector, the module with the next highest probability is activated. If all modules classify the *complementary class* the overall output is no activity and the

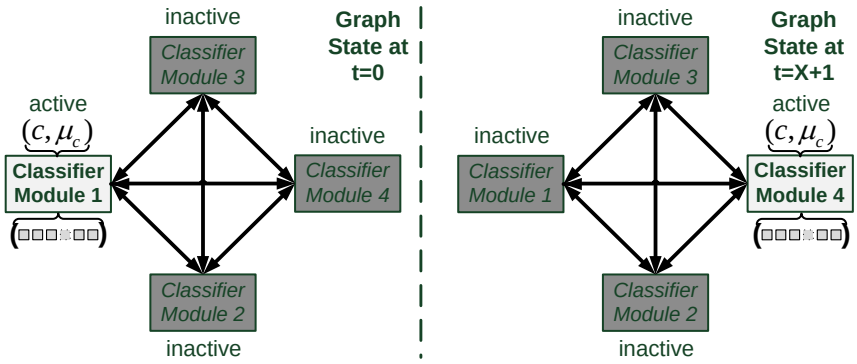


Figure 17: State of modular classifiers in graph at time $t = 0$ and $t = X + 1$

processing proceeds with the next feature vector. All the transition probabilities are saved in an adjacency matrix, which is described in the following:

$$A = \begin{pmatrix} \mathbb{P}(\mathbf{M}_1|\mathbf{M}_1) & \dots & \mathbb{P}(\mathbf{M}_j|\mathbf{M}_1) & \dots & \mathbb{P}(\mathbf{M}_m|\mathbf{M}_1) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbb{P}(\mathbf{M}_1|\mathbf{M}_i) & \dots & \mathbb{P}(\mathbf{M}_j|\mathbf{M}_i) & \dots & \mathbb{P}(\mathbf{M}_m|\mathbf{M}_i) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbb{P}(\mathbf{M}_1|\mathbf{M}_m) & \dots & \mathbb{P}(\mathbf{M}_j|\mathbf{M}_m) & \dots & \mathbb{P}(\mathbf{M}_m|\mathbf{M}_m) \end{pmatrix} \tag{27}$$

Since the recognition of the *complementary class* indicates a definite transition the probabilities of the next module being the same one is zero, which is described accordingly:

$$\mathbb{P}(\mathbf{M}_i|\mathbf{M}_i) = 0, \text{ with } i = 1, .., m$$

Since a transition from $\mathbf{M}_i$ to $\mathbf{M}_j$ is the same as from $\mathbf{M}_j$ to $\mathbf{M}_i$ the probability could be the same with $\mathbb{P}(\mathbf{M}_j|\mathbf{M}_i) = \mathbb{P}(\mathbf{M}_i|\mathbf{M}_j)$. Therefor the adjacency matrix would be symmetric with a main diagonal of zeros.

Adjacency Matrix Learning The transition probabilities in the adjacency matrix need somehow to be determined. Here the behavior of the previously described dynamic queue can provide this probabilities. In a training phase, the transition of the modules get recorded, where the probability $\mathbb{P}(\mathbf{M}_j|\mathbf{M}_i)$ is determined due to a transitions from module $\mathbf{M}_i$ to $\mathbf{M}_j$ for $i \neq j$. For a test data set based on normal human behavior, the total amount of transitions from the respective module is counted and then the number of transition from $\mathbf{M}_i$ to $\mathbf{M}_j$ is divided by it. This is a very simple approach, more sophisticated ones are imaginable. But, since the focus of this thesis lies on investigating the dynamic queue as scheduling technique, this is not further investigated.

With this kind of model learning there is a huge problem, since the probabilities are based on the transitions of a modular classifier, which is error prone itself. The question here is what part of the transition probability is based on faulty classifications and what actually reflects the human behavior. Here an actual test of the approach could give detailed insights.

Discussion and Conclusions This approach of organizing the module transition according to probabilities of actual human behavior has two big advantages: one is the reduced calculation effort and the other one is increased recognition accuracy. When the currently active module is recognizing the *complementary class* if the next module which is activated is the most likely one, no other module possibly needs to be activated on this feature vector. As the module transitions do not occur very often and the grouping of the classes per module is influential too, this effect could be marginal. The more modules that need to be evaluated until the correct one is found, the more errors of the modules need to be taken into account. Again, the module transitions are less likely than the classifications per active module, which also implies that the positive effect on the recognition accuracy will not be significant.

Although, the effect rises on both (calculation effort and accuracy) when the number of modules is increasing, fewer classes get recognized with each module.

More negative and limiting aspects for this approach could be found though. The probability model that is used for the module scheduling needs to be identified and modified when new modules are added. A method of training such a model upon a dynamic queue was introduced, but there are certain problems with that approach. The transitions of modules according to changes of human activities are captured in the probabilities, but also errors in classification and transition. If e.g. during the training phase a module gets activated very rarely, since this activity is not performed often, any transition to it would have a low probability. This circumstance can also occur when the module is rarely activated since another module is strongly interfering with it. This leads to a wrong module which influences then the performance of the activity recognition in a negative way. When the transition probabilities are determined on the base of a ground truth data set, the resulting model would than be more accurate, but also more effort by an expert is needed. Also, the ground truth is individual to the respective user and the resulting model does not have to be fitting for every other user. On every module that gets added to the activity recognition the adjacency matrix has to be extended, which again increases the effort for training or expert supervision input.

As can be seen, the usage of a probabilistic model to schedule the classifier modules raises a lot of additional questions. This fact and the fact that this approach is especially not **flexible** prevents it from being evaluated or further discussed in this thesis. The dynamic queue has definitely more advantages than the probabilistic model, thats why the modular activity recognition is only evaluated implementing the dynamic queue approach.

3.2.4 Other Classifier Module Scheduling Techniques

There are other methods that can additionally or instead be used for the module scheduling. Some of these approaches are briefly discussed in this subsection, but do not get evaluated or used further throughout this thesis.

Weighted Selection The idea behind the dynamic queue is, that modules organize themselves in the queue, such that the module with the highest possibility of being the next active one is close to the head of the queue. But due to errors in recognition, sometimes a module that falsely classifies the feature vectors is occurring in front of the correct one. This module would therefore classify all the feature vectors, until the error is eliminated or another more clearly distinguishable pattern occurs. As experience shows this effect can occur when the system is used in real practice.

A solution for this problem could be that the modules are not organized in a dynamic queue, but when the current active module is recognizing the *complementary class* all other modules become activated. If more than one module is recognizing a class besides the *complementary* one, the module with the highest reliability measure is chosen.

The problem with this approach is the increased calculation effort when transitions occur, since all modules need to be evaluated. In cases where this transitions happen rarely, the increase is not high, but this always depends on the users behavior and the grouping of classes per module. Again, this method would raise additional questions to be evaluated and the positive effect due to the modularity onto the challenge **resources** would be lowered. Therefore, this approach is not investigated further throughout this thesis.

Recurrent Transitions As for any classification, the classification on the *complementary class* is also error prone. In the case of a probabilistic model for module scheduling, the conditional probability of recurrent transitions was set to zero. It is not the only possibility for the recurrent transition to be impossible. A low pass filter could be included in the probabilistic model, so the first few transitions are not made onto one another, but onto the same module. The conditional probability $\mathbb{P}(\mathbf{M}_i|\mathbf{M}_i)$ for a recurrent transition could be set to a upper limit, which is higher than any probability $\mathbb{P}(\mathbf{M}_i|\mathbf{M}_i) > \mathbb{P}(\mathbf{M}_j|\mathbf{M}_i)$ for $i \neq j$. On every recurrent transition the probability $\mathbb{P}(\mathbf{M}_i|\mathbf{M}_i)$ is then lowered according to a constant or the reliability measure, so over time one of the non recurrent conditional probabilities is higher than the recurrent one $\mathbb{P}(\mathbf{M}_i|\mathbf{M}_i) < \mathbb{P}(\mathbf{M}_j|\mathbf{M}_i)$.

A probabilistic model like a Markov chain with recurrence for module scheduling is more sound than the proposed method. Since this approach is not further investigated, it is more of a suggestion based on the experience with modular activity recognition than a mathematically funded decision. The problems of the probability based module scheduling with extension are the same as without extension: how can the probabilistic model be determined and how can the model be extended?

Additional Periodic Scheduling The problem of the dynamic queue and the probabilistic module scheduling is always possible that they can end up in a faulty state. E.g. if a module is in front of another module in the dynamic queue and this module is falsely classifying the feature vectors intended for the other module, then there never will be a transition to the correct module. Due to the nature of the dynamic queue, this correct module could end up as the last module and this other module in front would always classify the feature vectors intended for the last module. For the probabilistic scheduling approach the problem is similar. If e.g. a module has with no other module a connection with the highest probability, the module is only activated if the transitions with higher probability are not taken due to *complementary class* detection. If in some cases one module with higher probability is often falsely classifying the data then the correct module will rarely be activated.

A solution to this problem could be that periodically a different scheduling technique is used instead of the regular one. E.g. periodically on a transition all other modules are activated in parallel and the one with the highest reliability of detecting a class besides the *complementary* one would stay active. With this method the case where the probabilistic model or the queue ends up in a faulty state is never permanent. Again, since the focus of this thesis is to understand the modularity and not the module scheduling, this extension of the scheduling is not investigated further.

Other Classifier Fusion Methods There are many more scheduling methods for the modular activity recognition possible than the ones listed thus far. As described in subsection 2.3 there are many methods for selection or fusion of classifiers. Some of these methods could be combined with the novel modular approach due to the *complementary class*. Further investigations on which methods could be applicable are part of future work and not of this thesis. All evaluation, analysis and implementation of the modular activity recognition which is following is based on the dynamic queue concept.

3.3 Modular Classifier Training Algorithm

The novel modular activity recognition presented in this thesis has several improvements over standard classifiers. One is the modularity itself and the recognition of a complementary class to detect module transitions. Another one is the recurrent edge in the mapping function, which enables the calculation of a reliability measure and makes the classification more robust. A third improvement is the usage of covariant memberships in the mapping function, which are especially suited for classification of highly correlated accelerometer measurements. A last improvement to be named here is the ability to personalize, generalize or just change classification modules temporarily without destroying the original mapping abilities permanently. All these improvements need especially to be considered in the training algorithm of the RFIS mapping function. For these reasons a novel machine learning algorithm was composed, which is presented in this subsection.

First, the training and check data is separated according to the modules the classification of this data is done with. Then the data for training the complementary class is composed. Also, the data needs to be randomized, so the classification accuracy can correctly be determined.

Second, the subtractive clustering is introduced, which is used for initialization of the Gath-Geva clustering. The subtractive clustering helps to cope with the problems of the Gath-Geva clustering, such as the initial cluster centers and the numerical instability.

Third, the Gath-Geva clustering is explained, which delivers the desired covariant cluster shapes. These clusters are corresponding to the used covariant memberships in the mapping function.

Fourth, a linear regression delivers the last components for constructing a TSK-FIS, the linear functional consequence. The least squares method is explained and how the parameters of the consequences can be calculated through the solution of an overdetermined linear equation.

Fifth, a genetic algorithm is presented to determine the optimal subset of cluster centers, which is used for initialization of the Gath-Geva clustering. The fitness values and the individuals genomes are defined. Also, the genetic algorithm with selection, recombination and mutation is explained.

Sixth, the overall training algorithm is explained, that connects all of the other methods together and results in the RFIS mapping functions used in the classification modules.

Last, the search for a bit vector is introduced, which is used for personalization, generalization and adaptation of the mapping functions. This step is additional to the common training, which is why it is explained last.

3.3.1 Data Grouping

The overall data is split into two pairs, the training and the check data. The training data is used to train the classification modules and the check data to control the outcome of the training. The check data especially can be used as termination criteria for the overall training algorithm. Good results could be archived with the check being one third the size of the training data. There will be more on this in the evaluation section.

Activity Grouping The annotated training and check data has to be grouped, to enable the training of a modular classifier structure. First, it has to be decided which module should classify which activities. Each module should not classify too many activity classes besides the *complementary* one, otherwise the advantages towards a monolithic approach are negated. A method that can be used to optimize the module alteration in the dynamic queue (subsec. 3.2.2) is based on a semantic grouping of the activity classes per module. Here the so called *conditional context* indicates which classes should be recognized with the same module.

Another method is model based, where the cost and accuracy of the resulting modular activity recognition indicates which activity classes should be grouped together. This model is set up in the subsection 4.4.1, where the opportunities of the modular activity recognition to cope with the challenge **resources** are discussed and analyzed. The model based approach is not investigated further or used in any system that is evaluated in this thesis. This is because the composition of activity classes according to a model is highly complex and would not give any insights into the modular activity recognition corresponding to the proposed challenges.

The simplest method that also results in the highest modularity is, when each module is only recognizing one activity class besides the *complementary* one. In this scenario no decision about the grouping has to be made, nor any model has to be established. The disadvantages of this approach could be, that the overhead due to the model switch and the computation of the *complementary* class could be higher compared to the other methods. Evaluations

should show whether this method or the others result in the least complex and most accurate activity recognition. In the end there is always the possibility to randomly group the activities, but this method is not considered to have good results.

Classes per Module Grouping In this manner, the training and check data are split up according to the annotation. Each module is trained separately, so the training of module $\mathbf{M}_i$ is done on the training data $\mathcal{V}_i^{tr}$, whose designated output $\overrightarrow{\mathbf{u}}_i$ only consist of classes $c_{ij} \in \mathcal{C}_i$ native to this module and the same applies for the grouping of the check data.

Data for Complementary Class Training To train the modular classifiers, they need to be trained on both, their classes $c_{ij} \in \mathcal{C}_i$ and on the complementary class $\overline{c}_i$. The training $\mathcal{V}_i^{tr}$ and check data sets $\mathcal{V}_i^{ck}$ for a classifier module $\mathbf{M}_i$ are unified with a selection of input data pairs of all other classifiers $\mathcal{V}_i^{\overline{c}} \subset \bigcup_{k \neq i} \mathcal{V}_k^{tr}$. This selection is labeled zero – which indicates the complementary class $\overline{c}_i$ in every module $\mathbf{M}_i$ – and added to the normal training and check data sets of this classifier. The actual training and check data is therefore $\mathcal{V}_i^{tr \cup \overline{c}}$ and $\mathcal{V}_i^{ck \cup \overline{c}}$, which are called $\mathcal{V}_i^{tr}$ and $\mathcal{V}_i^{ck}$ throughout the rest of this thesis for reasons of simplicity. If the data is grouped this way, then the next step of the training algorithm can begin, the initial subtractive clustering.

3.3.2 Subtractive Clustering

Since the Gath-Geva clustering is numerically instable, the number of cluster centers is needed in advance and often does not converge to valid cluster centers with random initialization, a clustering is needed that can be used for initialization. To identify the initial cluster centers and the amount of clusters for the Gath-Geva clustering a subtractive clustering [42] is used. The subtractive clustering does not need the number of clusters in advance. Also, there are not much parameter to be set and the algorithm works very well if they do not vary much from the ones suggested by Chiu [41]. The biggest advantage of subtractive clustering above e.g. mountain clustering [153] is, that it does not rely on a grid width. With subtractive clustering every vector of the data to be clustered is considered as a potential cluster center. Therefore for every data point the potential of being a cluster center needs to be calculated, which results in a high calculation complexity with rising cluster data length. The complexity and the inability of adapting to covariant cluster shapes are the only known disadvantages.

The subtractive clustering is done on the class specific training data $\mathcal{V}_{c_{kl}}^{tr} \subseteq \mathcal{V}_k^{tr}$ with $c_{kl} \in \mathcal{C}_k$ so the clusters imply rules for only one class. This enhances the interpretability of the resulting FIS and increases the accuracy of the classification process due to rules specific to a class. Since the description of the subtractive clustering algorithm does not depend on this distinction and the illustration would become more complicated, the algorithm is described along a general set of training data $\mathcal{V}^{tr}$ and the same applies for the later explained Gath-Geva clustering.

Cluster Potential The subtractive clustering starts with the calculation of a potential P_i for each of the training data vectors $\overrightarrow{\mathbf{v}}_i \in \mathcal{V}^{tr}$ to be a cluster center, which is done accordingly:

$$P_i(\mathcal{V}^{tr}) = \sum_{i=1}^{n} e^{-\alpha ||\overrightarrow{\mathbf{v}}_i - \overrightarrow{\mathbf{v}}_j||_2^2} \text{, with } \alpha = \frac{4}{r_a^2} \tag{28}$$

The Euclidean distance is used to calculate the distance of the vector $\overrightarrow{\mathbf{v}}_i$ to all other vectors $\overrightarrow{\mathbf{v}}_j$ with $j \neq i$ in the training data set $\mathcal{V}^{tr}$. The distance is the negative exponent of Euler's constant and multiplied with a factor α. Alpha consists of one of the algorithms parameters, the radius of acceptance r_a. The sum of all exponentiated distances is summed up and results in the potential P_i.

The training data vector $\overrightarrow{\mathbf{v}}_i$ with the highest potential P_i becomes the first cluster center $\overrightarrow{\mathbf{v}}_1^*$. All cluster centers are added to an initially empty set $\mathcal{V}^C = \emptyset$ of cluster centers $\mathcal{V}^C \Leftarrow \mathcal{V}^C \cup \{\overrightarrow{\mathbf{v}}_1^*\}$.

Potential Update With the first cluster center the potentials P_j for $j \neq i$ are updated in the following manner:

$$P_i \Leftarrow P_i - P_1^* \cdot e^{-\beta ||\overrightarrow{\mathbf{v}}_i - \overrightarrow{\mathbf{v}}_1^*||_2^2} \text{, with } \beta = \frac{4}{r_b^2} \tag{29}$$

Here β is a new parameter that is determined through the radius r_b. The closer a data vector $\overrightarrow{\mathbf{v}}_i$ is to the cluster center $\overrightarrow{\mathbf{v}}_1^*$ the more the potential P_i is lowered according to the radius r_b. To prevent closely spaced clusters a value somewhat greater than r_a is suggested by Chiu [42]. He also mentioned $r_b = 1.25r_a$ to be a good choice.

Again, the training data vector $\overrightarrow{\mathbf{v}}_k$ with the highest potential P_k gets added to the set of $\mathcal{V}^C \Leftarrow \mathcal{V}^C \cup \{\overrightarrow{\mathbf{v}}_k^*\}$ and the potentials P_i of all other training data vectors $\overrightarrow{\mathbf{v}}_i \in \mathcal{V}^{tr}$ get updated accordingly:

$$P_i \Leftarrow P_i - P_k^* \cdot e^{-\beta||\overrightarrow{\mathbf{v}}_i - \overrightarrow{\mathbf{v}}_k^*||_2^2} \tag{30}$$

Stop Criterion and Extended Selection Algorithm The process of acquiring new cluster centers and updating the potentials is repeated until the potential P_k^* of a new cluster center $\overrightarrow{\mathbf{v}}_k^*$ is only a certain fraction of the first cluster centers potential P_1^*. Another method which was used in this thesis is the algorithm 1 for stopping the process, accepting or rejecting new cluster centers. In the algorithm two new parameters are introduced, the accept $\bar{\epsilon}$ and the

Algorithm 1 Subtractive clustering - accepting and rejecting new cluster centers

```
if P_k^* > ε̄ P_1^* then
    V^C ⇐ V^C ∪ {v_k^*}
else if P_k^* < ε̲ P_1^* then
    reject v_k^*
    exit
else
    δ_min = min_i ||v_k^* − v_i^*||_2
    if δ_min / r_a + P_k^* / P_1^* ≥ 1 then
        V^C ⇐ V^C ∪ {v_k^*}
    else
        reject v_k^*
        for i ∈ V^tr do
            v_tmp ⇐ v_k^*
            if v_tmp == v_i then
                P_i ⇐ 0
            else if max_i |P_k^* − P_i| and v_i ∉ V^C then
                v_k^* ⇐ v_i
                P_k^* ⇐ P_i
            end if
        end for
        retest for P_k^*
    end if
end if
```

reject $\underline{\epsilon}$ ratio. All potential cluster centers $\overrightarrow{\mathbf{v}}_k^*$ with a potential P_k^* above $\bar{\epsilon}$ are automatically accepted, those below $\underline{\epsilon}$ are rejected and the subtractive clustering stops. The vectors $\overrightarrow{\mathbf{v}}_k^*$ with a potential P_k^* in between the two ratios get tested further according to their Eucledian distance δ_{min} to the next neighboring cluster center. If the distance is in a certain range, the vector again gets accepted and is added to the set of cluster centers $\mathcal{V}^C \Leftarrow \mathcal{V}^C \cup \{\overrightarrow{\mathbf{v}}_k^*\}$. If not, its potential is set to zero $P_i \Leftarrow 0$ and the next vector to be tested is selected according to the next highest potential to the previously tested one.

The subtractive clustering provides a set of cluster centers $\mathcal{V}^C = \{\overrightarrow{\mathbf{v}}_1^*, .., \overrightarrow{\mathbf{v}}_{n_C}^*\}$ and an upper limit for the amount of clusters n_C. Since with covariant cluster shapes for the highly correlated sensor data the number of clusters towards a multivariate clustering is lowered (e.g. fig. 9), the amount of cluster centers n_C provided by the subtractive clustering is only an upper limit. To find the optimal number of initial cluster centers $\mathcal{V}^{C_I} \subseteq \mathcal{V}^C$ a genetic algorithm is used, which will be explained in subsection 3.3.5 and the overall selection algorithm in3.3.6.

3.3.3 Gath-Geva Clustering

A Gath-Geva [57] clustering is used to determine the membership functions 10 used in the RFIS mapping function 13, which is explained in subsection 3.1.3. It is also indirectly used to determine the amount of rules 11, since the number of clusters is the same. The subtractive clustering delivers the upper bound for the amount of clusters n_C, from which with a genetic algorithm successively is searched down until a TSK-FIS with maximum recognition accuracy results, but this is explained later.

The Gath-Geva clustering has certain disadvantages, such as numerical instability and adding that with random cluster center initialization the algorithm might not be able to converge to the actual ones. This disadvantages can be coped with due to a initialization with a different clustering algorithm. Here a subtractive clustering is used to provide the initial cluster centers and an upper bound on the number of clusters. The biggest advantage of the Gath-Geva clustering is, that it provides cluster shapes which fit onto correlated data (fig. 9, right). These clusters can then be translated in desired covariant membership functions (eqn. 10). Certain extensions of the Gustafson-Kessel clustering [65] could deliver similar cluster shapes, but since the Gath-Geva clustering natively providing the desired covariant Gaussian functions it is preferred.

Gath-Geva Clustering Gath and Geva [57] generalize the maximum likelihood estimation for the fuzzy clustering. They assume, that the normal distribution N_i, with the expected value for the cluster center $\overrightarrow{\mathbf{v}}_i^*$, the covariance matrix Σ_i, and the a-priory probability P_i are used to generate the data $\overrightarrow{\mathbf{v}}_j \in \mathcal{V}^{tr}$. For initialization of the clustering, a set of cluster centers $\mathcal{V}^{C_I}$ needs to be estimated, in this case a subset $\mathcal{V}^{C_I} \subseteq \mathcal{V}^C$ of the centers found through the subtractive clustering is used. The number of clusters has an upper bound n_C found through the subtractive clustering. The initial memberships are calculated according to the fuzzy c-means clustering [34] through a Euclidian distance, as follows:

$$\mu_{ij}^{(1)} = \frac{1}{\|\overrightarrow{\mathbf{v}}_j - \overrightarrow{\mathbf{v}}_i^{*(1)}\|_2} \tag{31}$$

In set $\mathcal{V}^{C_I} = \{.., \overrightarrow{\mathbf{v}}_i^{*(l)}, ..\}$ the index (l) is introduced, which denotes the cluster center vector $\overrightarrow{\mathbf{v}}_i^{*(l)}$ of iteration l. The initial set of cluster centers $\mathcal{V}^{C_I}$ the vectors $\overrightarrow{\mathbf{v}}_i^{*(1)}$ have the iteration number (1).

In further iterations $l = 2, 3..$ of the algorithm, the membership is calculated according to the following equation:

$$\mu_{ij}^{(l)} = \frac{1}{\sum_{k=1}^{n_C} \left(\frac{D_{ij}(\overrightarrow{\mathbf{v}}_j, \overrightarrow{\mathbf{v}}_i^*)}{D_{kj}(\overrightarrow{\mathbf{v}}_j, \overrightarrow{\mathbf{v}}_k^*)} \right)^{\frac{2}{n_C - 1}}} \tag{32}$$

The first step of the clustering algorithm is to calculate the cluster centers $\overrightarrow{\mathbf{v}}_i^*$, as follows:

$$\overrightarrow{\mathbf{v}}_i^{*(l)} = \frac{\sum_{j=1}^{n_C} \mu_{ij}^{(l-1)} \overrightarrow{\mathbf{v}}_j}{\sum_{t=1}^{n_C} \mu_{ij}^{(l-1)}} \, , 1 \leq i \leq n_C \tag{33}$$

The second step is to determine the covariance matrix, which is needed to calculate the distance measurement. The covariance matrix is calculated as follows:

$$\Sigma_i^{(l)} = \frac{\sum_{j=1}^{n_{tr}} (\mu_{ij}^{(l-1)})^2 (\overrightarrow{\mathbf{v}}_j - \overrightarrow{\mathbf{v}}_j^{*(l)})^T (\overrightarrow{\mathbf{v}}_j - \overrightarrow{\mathbf{v}}_j^{*(l)})}{\sum_{j=1}^{n_{tr}} (\mu_{ij}^{(l-1)})^2} \tag{34}$$

The algorithm employs a distance measurement based on the fuzzy maximum likelihood estimates, proposed by Bezdek and Dunn [33]:

$$D_{ij}(\overrightarrow{\mathbf{v}}_j, \overrightarrow{\mathbf{v}}_i^*) = \frac{\sqrt{|\Sigma_i|}}{P_i^{(l)}} e^{-\frac{1}{2}(\overrightarrow{\mathbf{v}}_j - \overrightarrow{\mathbf{v}}_i^{*(l)}) \Sigma_i^{-1} (\overrightarrow{\mathbf{v}}_i - \overrightarrow{\mathbf{v}}_i^{*(l)})^T} \tag{35}$$

The a-priory probability is:

$$P_i^{(l)} = \frac{\sum_{j=1}^{n_{tr}} \mu_{ij}^{(l-1)}}{\sum_{j=1}^{n_{tr}} \sum_{k=1}^{n_C} \mu_{kj}^{(l-1)}} \tag{36}$$

The steps 1 and 2 of the algorithm get now repeated until iteration l is reached where $\sum_{i=1}^{n_C} \|\overrightarrow{\mathbf{v}}_i^{*(l)} - \overrightarrow{\mathbf{v}}_i^{*(l-1)}\|_2 < \epsilon$.

Gath-Geva Algorithm In more detail the Gath-Geva clustering is described in the algorithm 2.

Algorithm 2 Gath-Geva Clustering

for $i := 1$ **to** n_C **do**
  for $j := 1$ **to** n_{tr} **do**
    $\mu_{ij}^{(1)} = \frac{1}{\|\overrightarrow{\mathbf{v}}_j - \overrightarrow{\mathbf{v}}_i^{*(1)}\|_2}$
  end for
end for
repeat
  $l \Leftarrow l + 1$
  for $i := 1$ **to** n_C **do**
    $\overrightarrow{\mathbf{v}}_i^{*(l)} = \frac{\sum_{j=1}^{n_C} \mu_i^{(l-1)}(\overrightarrow{\mathbf{v}}_j)\overrightarrow{\mathbf{v}}_j}{\sum_{t=1}^{n_C}(\mu_{ij}^{(l-1)})^2}$
    $\Sigma_i^{(l)} = \frac{\sum_{j=1}^{m}(\mu_{ij}^{(l-1)})^2(\overrightarrow{\mathbf{v}}_j - \overrightarrow{\mathbf{v}}_j^{*(l)})^T(\overrightarrow{\mathbf{v}}_j - \overrightarrow{\mathbf{v}}_j^{*(l)})}{\sum_{j=1}^{m}(\mu_{ij}^{(l-1)})^2}$
    for $j := 1$ **to** n_{tr} **do**
      $D_{ij}(\overrightarrow{\mathbf{v}}_j, \overrightarrow{\mathbf{v}}_i^*) = \frac{\sqrt{|\Sigma_i^{(l)}|}}{P_i^{(l)}} e^{-\frac{1}{2}(\overrightarrow{\mathbf{v}}_j - \overrightarrow{\mathbf{v}}_i^{*(l)})\Sigma_i^{(l)-1}(\overrightarrow{\mathbf{v}}_i - \overrightarrow{\mathbf{v}}_i^{*(l)})^T}$
    end for
  end for
  for $i := 1$ **to** n_C **do**
    for $j := 1$ **to** n_{tr} **do**
      $\mu_{ij}^{(l)} = \frac{1}{\sum_{k=1}^{n_C}\left(\frac{D_{ij}(\overrightarrow{\mathbf{v}}_j, \overrightarrow{\mathbf{v}}_i^*)}{D_{kj}(\overrightarrow{\mathbf{v}}_j, \overrightarrow{\mathbf{v}}_k^*)}\right)^{\frac{2}{n_C-1}}}$
    end for
  end for
until $\left(\sum_{i=1}^{n_C} \|\overrightarrow{\mathbf{v}}_i^{*(l)} - \overrightarrow{\mathbf{v}}_i^{*(l-1)}\|_2\right) < \epsilon$

Gath-Geva Issues and Strengths The Gath-Geva clustering has certain problems, e.g. it can overflow at large n_C values due to the inversion problems. Also, it might not converge to the right cluster centers if the initial cluster centers are randomly initialized due to the exponential distance norm. Another issue is, that the algorithm cannot determine the number of clusters itself. This is why a subtractive clustering is used to provide the initial cluster centers and the number of clusters.

On the other side the Gath-Geva clustering can adapt to varying cluster shapes, densities and sizes. The clustering delivers the covariance matrices and the cluster centers for the membership functions of the RFIS mapping used in the modular activity classification. The covariance matrices $\Sigma_i^{(l)}$ and cluster center vectors $\overrightarrow{\mathbf{v}}_i^{*(l)}$ of the last iteration of the algorithm are used in the classification. To fully construct a TSK-FIS furthermore the linear functional consequences needs to be determined, which is explained in the next subsection.

3.3.4 Linear Regression

On basis of the determined clusters, which are specified through the membership $\mu_i(\overrightarrow{\mathbf{v}}_j)$ (eqn. 10), and the number of clusters n_{C_I} the parameters $a_{1i}, .., a_{(n+1)i}$ of the linear functional consequences $f_i(\overrightarrow{\mathbf{v}}_j)$ (eqn. 5) for $i = 1, .., m$ can be calculated. The number of rules m is the number of clusters n_{C_I}. The non-linear membership functions μ_i imply the linear consequences f_i according to the determined output $\overrightarrow{\mathbf{u}}$ for the training data $\mathcal{V}^{tr}$, which results in a linear equation. The parameters a_{ki} of the linear functional consequence f_i are calculated by solving this linear equation. The solution is a linear regression approach which is called least squares method [7].

Error of Mapping For each rule the consequence parameters are calculated separately by minimizing the mean squared error:

$$\mathcal{E}(\mathcal{V}^{tr}) = \sum_{j=1}^{n_{tr}} (\varepsilon(\overrightarrow{\mathbf{v}}_j)))^2 = \sum_{j=1}^{n_{tr}} (u_j - \mathbf{S}(\overrightarrow{\mathbf{v}}_j))^2 = \sum_{j=1}^{n_{tr}} (u_j - \sum_{k=1}^{(n+1)\cdot m} b_{jk} a_k)^2 \tag{37}$$

Here $\overrightarrow{\mathbf{u}} = (.., u_j, ..)$ is the desired output for input vector $\overrightarrow{\mathbf{v}}_j \in \mathcal{V}^{tr}$. The parameters a_{ki} map on the values a_k of vector $\overrightarrow{\mathbf{a}}$, as follows:

$$\begin{array}{rccc} & \text{consequence} & & \text{consequence} \\ & \text{param. rule 1} & & \text{param. rule m} \\ \overrightarrow{\mathbf{a}} = & (\overbrace{a_{11}, .., a_{(n+1)1}}, & .. & , \overbrace{a_{1m}, .., a_{(n+1)m}}) \\ = & (a_1, .., a_{(n+1)}, & .. & , a_{((n+2)\cdot(m-1))}, .., a_{((n+1)\cdot m)}) \, , \end{array} \tag{38}$$

where $k = 1, .., (n+1), .., (n+1)\cdot m$ is the index of the vector $\overrightarrow{\mathbf{a}}$ is. Also, the values b_{jk} correspond to the original TSK-FIS $\mathbf{S}(\overrightarrow{\mathbf{v}}_t)$ in the following way:

$$b_{jk} = \frac{\mu_i(\overrightarrow{\mathbf{v}}_j)}{\sum_{j=1}^{m} \mu_i(\overrightarrow{\mathbf{v}}_j)} v_{kj} \, , \tag{39}$$

with $v_{((n+1)\cdot l)j} = 1$ and $l = 1, .., m$. The values b_{jk} are listed in matrix $B = (b_{jk})$ and $\overrightarrow{\mathbf{v}}_j = (.., v_{kj}, ..)$.

Error Gradient The error gets minimized for each parameter of the consequence by setting its gradient to zero:

$$\frac{\partial \mathcal{E}}{\partial a_l} = \sum_{j=1}^{n_{tr}} \left(2(u_t - \sum_{k=1}^{(n+1)\cdot m} b_{jk} a_k)(-b_{tl}) \right) = 0 \tag{40}$$

Solving the equation 40 for all parameters a_k leads to the following linear equation:

$$\begin{array}{rcl} 2(B\overrightarrow{\mathbf{a}}^T - \overrightarrow{\mathbf{u}})^T B & = & 0 \\ B^T(B\overrightarrow{\mathbf{a}}^T - \overrightarrow{\mathbf{u}}) & = & 0 \\ B^T B\overrightarrow{\mathbf{a}}^T & = & B^T\overrightarrow{\mathbf{u}} \\ \overrightarrow{\mathbf{a}}^T & = & \left((B^T B)^{-1} B^T\right) \overrightarrow{\mathbf{u}} \end{array} \tag{41}$$

Singular Value Decomposition (SVD) SVD is used to decompose the matrix B into three matrices with certain attributes. The matrix $B = UVD^T$ is decomposed in such a way that the multiplication of the matrix U with its transposed U^T is the unity matrix. Also, D is a diagonal matrix and V is orthogonal. This results in a new linear equation for equation 41, which structures accordingly:

$$\begin{array}{rcll} \overrightarrow{\mathbf{a}}^T & = & (VD^T U^T UDV^T)^{-1} VD^T U^T \overrightarrow{\mathbf{u}} & (B_T = \mathcal{U}\mathcal{D}\mathcal{V}^T) \\ & = & (VD^T DV^T)^{-1} VD^T U^T \overrightarrow{\mathbf{u}} & (U^T U = \mathcal{E}) \\ & = & \left((D^T D)VV^T\right)^{-1} VD^T U^T \overrightarrow{\mathbf{u}} & (D^T D \text{ diagonal}) \\ & = & (D^T D)^{-1} VD^T U^T \overrightarrow{\mathbf{u}} & (V \text{ orthogonal } \Rightarrow VV^T = \mathcal{E}) \\ & = & V(D^T D)^{-1} D^T U^T \overrightarrow{\mathbf{u}} & \\ & = & VD^{-1}(D^T)^{-1} D^T U^T \overrightarrow{\mathbf{u}} & \\ & = & VD^{-1} U^T \overrightarrow{\mathbf{u}} & \end{array} \tag{42}$$

The SVD can be calculated with a QR algorithm. Also, the SVD can help solving the equation 41, since the solution of over determined linear equations is mostly numerically instable.

After the linear regression the TSK-FIS can be constructed. The least squares method delivered the last component of the mapping function, the linear functional consequence 5. Before, the Gath-Geva clustering delivered the covariant Gaussian membership functions 10, where the results of a subtractive clustering are used for initialization. In the descriptions of the machine learning algorithms up to this point certain singularities special to a modular recurrent classification have not been mentioned. This is because the algorithms have a certain complexity already and the description of them along with the modularity and the recurrence would unnecessarily complicate the illustration. The details of the overall novel algorithm, along with the specifications for the modularity and recurrence, are described later in this subsection.

3.3.5 Genetic Algorithm

Genetic algorithms belong to the group of evolutionary methods. Originally, they were devised for optimization with the ability to solve non-linear non-quadratic optimization problems [49][61].

The subtractive clustering only delivers an upper bound for the amount of cluster centers, since with the Gath-Geva clustering a better adaptation on correlated sensor data with less clusters is possible (see fig 9). The optimal number of clusters for the Gath-Geva clustering is not known in advance, thats why a top-down approach is presented, with which the amount of clusters according to the performance of the resulting TSK-FIS is determined. This best initialization of the Gath-Geva clustering is searched for with a genetic algorithm.

A complete search of the best subset of cluster centers handed over by the subtractive clustering would be possible, but with a high number of clusters the determination of the best subset can take a long time. In practice the full search turned out to be too time costly, since the Gath-Geva clustering on each subset of cluster centers needs to be done. Every clustering leads to a new identification of linear consequences through linear regression. Also, the accuracy of the mapping needs to be calculated on the whole training data set. So, in between the two clustering methods a genetic algorithm intercepts, which heuristically searches for the subset of initial cluster centers which results in the best classifier.

Fitness Value The genetic algorithm needs a fitness value which it can optimize the population of individuals upon. for this there are two possibilities: the percentage of correct classifications or the mean quadratic error. The optimization on the correct classifications percentage leads to a statistically better classifier. Since in the novel modular activity recognition architecture presented here, the filtering on a reliability measure results in the highest accuracy, the expressiveness of this value is important to the training algorithm, too. The mean quadratic error as fitness value results in a more expressive reliability measure. Experiments will show (see subsection 4.3) if the statistical classification accuracy with a most expressive reliability value is higher than one with a mapping function optimization directly on the classification percentage. At this point the mean quadratic error is considered to be the best choice as fitness value the genetic algorithm optimizes the mapping function upon.

The mean quadratic error $\mathcal{E}$ on the training data $\mathcal{V}^{tr}$ with the designated output $\overrightarrow{\mathbf{u}} = \{.., u_j, ..\}$ is calculated accordingly:

$$\mathcal{E}^{(l)}(\mathcal{V}^{tr}) = \sum_{j=1}^{n_{tr}} (u_j - \mathbf{S}^{(l)}(\overrightarrow{\mathbf{v}}_j))^2 \tag{43}$$

The equation 43 corresponds to equation 37, but is here specified on the needs for a fitness value. Since the optimization results in changes to the mapping function $\mathbf{S}$, an index (l) is identifying the current individual.

Most implementations of genetic algorithms optimizing on highest possible fitness value, so is the one used here. Therefore the fitness value function $\mathcal{F}^{(l)}$ is the inversion of the mean quadratic error of equation 43, which is shown in the following:

$$\mathcal{F}^{(l)}(\mathcal{V}^{tr}) = 1 - \frac{\mathcal{E}^{(l)}(\mathcal{V}^{tr})}{\max_k(\mathcal{E}^{(k)}(\mathcal{V}^{tr}))}$$

Here a problem arises, because minimum and maximum of the mean quadratic error is not known in advance. An upper limit needs to be found, that can be used instead of the unknown maximum. Since a genetic algorithm is used to find the optimal subset $\mathcal{V}^{C_I}$ of initial cluster centers for the Gath-Geva clustering, the whole set $\mathcal{V}^{C_I} = \mathcal{V}^{C}$ can be a starting point for the optimization. If the genetic algorithm cannot find a better or equally good subset $\mathcal{V}^{C_I}$, the whole set $\mathcal{V}^{C}$ of cluster centers provided by the subtractive clustering is used for initialization. The resulting mapping function $\mathbf{S}^{(0)}$ for the whole set $\mathcal{V}^{C}$ is defined with $l = 0$, which results in the final fitness value function accordingly:

$$\mathcal{F}^{(l)}(\mathcal{V}^{tr}) = \max\left(0, 1 - \frac{\mathcal{E}^{(l)}(\mathcal{V}^{tr})}{\mathcal{E}^{(0)}(\mathcal{V}^{tr})}\right) \tag{44}$$

Also, most of the genetic algorithm implementations can only search on positive fitness values, thats why the minimum is limited to zero. If in the hierarchical optimization process the best subset $\mathcal{V}^{C_I} \subset \mathcal{V}^{C}$ is only as good as the full set $\mathcal{V}^{C}$ of initial cluster centers, then the smallest subset is used since this results in less complexity of the classification.

Individuals Genome The genetic algorithm works on a population of individuals. Each individual is identically identified through a genome. The genome specifies the subset $\mathcal{V}^{C_I} \subset \mathcal{V}^C$ of initial cluster centers for the Gath-Geva clustering and therefore indirectly on the resulting mapping function $\mathbf{S}^{(l)}$. The genomes are defined accordingly:

$$\begin{array}{rcl} & & \overbrace{\qquad\qquad}^{\text{cluster centers for class } c_1} \qquad\qquad \overbrace{\qquad\qquad}^{\text{cluster centers for class } c_o} \\ \overrightarrow{\mathbf{g}}^{(l)} & = & (1, \quad .. \quad 1, \quad .. \quad 1, \quad .. \quad ,1) \\ \mathcal{V}^C & = & \{\overrightarrow{\mathbf{v}}^*_1, \quad .. \quad \overrightarrow{\mathbf{v}}^*_i, \quad .. \quad \overrightarrow{\mathbf{v}}^*_j, \quad .. \quad ,\overrightarrow{\mathbf{v}}^*_{nC}\} \end{array} \tag{45}$$

Each bit of the genome identifies a cluster center, where the cluster center $\overrightarrow{\mathbf{v}}^*_i$ is included in the subset $\mathcal{V}^{C_I}$ if the respective bit is one.

Since the clustering is done on each of a class specific training data separately, each cluster center can be assigned to a corresponding class c_i. Out of this results the boundary condition, that not all of the bits for one class can be zero, otherwise there would be no rule in the resulting TSK-FIS mapping function for this class. The class would therefore not exist in the mapping and the classification of this class would be attributed to incorrect classifications or coincidence.

Genetic Algorithm The aim is to find a maximum for a fitness value function through the modification the individuals, which are specified through their genomes. From each generation a subset of the individuals (from a population) with the best fitness are selected for the next generation. A subset of this subset gets mutated and recombined with a constant the size of the population. The bigger the search space, the more individuals are needed.

The recombination of two individual's genomes is done bitwise, where a fraction of each parent's genome is combined together in the offspring's genome. The recombination is explained along the following example:

$$\begin{array}{rcl} \overrightarrow{\mathbf{g}}^{(i)} & = & (1, \quad 0, \quad 0, \quad 1, \quad 1, \quad 1, \quad 0, \quad 1) \\ \overrightarrow{\mathbf{g}}^{(j)} & = & (0, \quad 1, \quad 1, \quad 1, \quad 0, \quad 1, \quad 1, \quad 1) \\ \overrightarrow{\mathbf{g}}^{(i\otimes j)} & = & (1, \quad 0, \quad 0, \quad 1, \quad 0, \quad 1, \quad 1, \quad 1) \end{array}$$

The fractions of each genome which the child inherits doeas not have to be the same size as shown in the example. Especially when the genomes are odd in size, one parent will have to administer more than the other. The cross product of the two parents genomes $\overrightarrow{\mathbf{g}}^{(i)}$ and $\overrightarrow{\mathbf{g}}^{(j)}$ is denoted with the binary operator $\otimes$. The operator is not deterministic if the fraction of each parent which are used in the child's genome are of random or probability distributed size.

Also, some of the offspring are mutated which is explained through the following example:

$$\begin{array}{rcl} \overrightarrow{\mathbf{g}}^{(i\otimes j)} & = & (1, \quad 0, \quad 0, \quad 1, \quad 0, \quad 1, \quad 1, \quad 1) \\ \overrightarrow{\mathbf{g}}^{(\oslash(i\otimes j))} & = & (1, \quad 1, \quad 0, \quad 1, \quad 0, \quad 0, \quad 1, \quad 1) \end{array}$$

Through the mutation a number of randomly selected bits are flipped in the offspring's genome. How many bits are flipped is determined according to a probability distribution. The mutations of the individuals are within a certain hamming distance to the original genome. The mutation operator $\oslash$ is again not a deterministic operation.

Since the recombination and the mutation of the genomes are no deterministic operations, each run of a genetic algorithm has a different outcome. Also, the initialization of the algorithm starts with a random determination of the individuals.

The parameters of a genetic algorithm are the population size, the boundary conditions and the stop criterion. There are more parameters to the most implementations of genetic algorithms, but these three are the most important ones in this case. For all other parameters standard values are used. The algorithms stop criterion is when for a certain number of generations the fitness of the best individual no longer improves. Since many individuals could have the same fitness value and certain combinations are prohibited through the boundary condition, the number of generations where no improvement leads to a stop of the algorithm is set to a low value. More on this conditions is presented in the next subsection, where the whole training algorithm for determining the RFIS mapping functions is described.

3.3.6 Training Algorithm

The overall training algorithm of the RFIS mapping function is the real novelty here. The combination of some of the algorithms to train a FIS has been presented so far. But, e.g. the ANFIS method [81] uses a gradient descent method to train a TSK-FIS, which is extremely calculation-intensive and does not support the use of covariant membership functions or recurrent edges. Therefore, the ANFIS approach is not applicable in this case. The novelty presented here is the combination of the different methods to train a Recurrent Fuzzy Inference System (RFIS) which is used in a modular activity recognition architecture. The algorithm is displayed in figure 18 and described in the following.

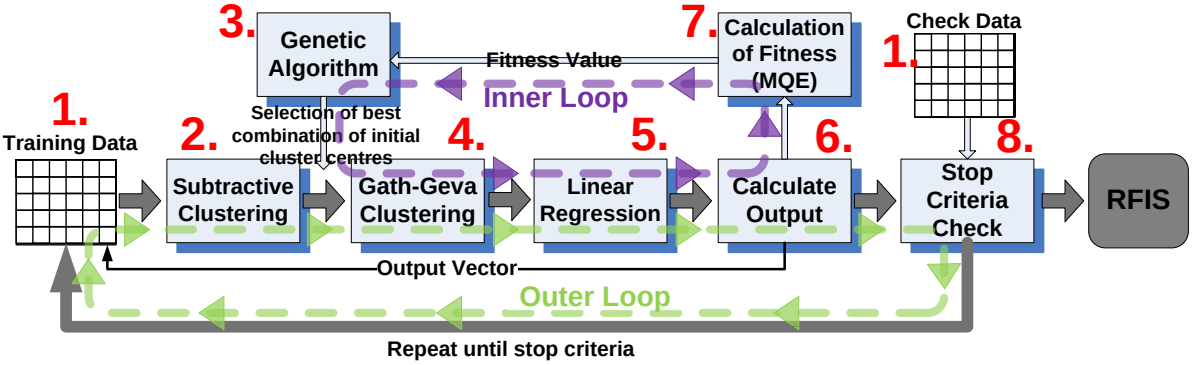


Figure 18: Training algorithm for RFIS mapping functions.

1. **Initialization:** The algorithm starts with the training and check data preparation. The whole set of annotated and preprocessed sensor data gets separated in training and check data. The size of the check data is about one third of the training data. Each classification module is trained separately, so the training data has to be split up. Also, the data for training on the complementary class needs to be prepared. The training data for the complementary class $\overline{c}_i$ of module $\mathbf{M}_i$ consists of training data of all the other modules M_j with $i \neq j$. Same applies for the preparation of the check data. Also the training and check data needs to be randomized in slices of width d, so the performance of the recurrent mapping function can be evaluated correctly.

   Outer Loop: With the addition of the output to the training and check data, the data needs again to be processed.
   Input: Annotated preprocessed (feature extraction) sensor data $\mathcal{V}$. In further iterations > 0 the RFIS output $\mathbf{S}_i(\mathcal{V}^{tr \cup \overline{c}_i})$ is added to the training data as new dimension.
   Output: Training $\mathcal{V}_i^{tr \cup \overline{c}_i}$ and check data $\mathcal{V}_i^{ck \cup \overline{c}_i}$ for each module $\mathbf{M}_i$ with $i = 1, .., n_M$ which includes data for training on the complementary class.

2. **Subtractive Clustering:** The training starts with a subtractive clustering, which is used to cope with the issues of the Gath-Geva clustering. The subtractive clustering uses multivariate cluster shapes and does not need the number of clusters in advance. It is also a numerically stable clustering algorithm. The cluster centers it determines are input for the Gath-Geva clustering. Also, the number of clusters are the upper bound for the Gath-Geva clustering.

   Input: Training data $\mathcal{V}_i^{tr \cup \overline{c}_i}$ for the training of module $\mathbf{M}_i$.
   Output: A set of cluster centers $\mathcal{V}_i^C$ and an upper bound n_C.

3. **Genetic Algorithm Interception:** Inbetween the subtractive clustering and the Gath-Geva clustering a genetic algorithm intercepts, which is used to determine the optimal subset of cluster centers $\mathcal{V}_i^{C_I}$. The subset is

used for the initialization of the Gath-Geva clustering. The fitness on which the genetic algorithm optimizes the subset is determined on the mean quadratic error which the resulting mapping function has on the training data $\mathcal{V}_i^{tr\cup\overline{c}_i}$.

Inner Loop: After the fitness value is determined on every individual in the population, the genetic algorithm determines the population for the next epoch.
Input: A set of cluster centers $\mathcal{V}_i^C$ and an upper bound n_C.
Output: A subset of cluster centers $\mathcal{V}_i^{C_I}$ and the actual number of clusters n_{C_I}.

4. **Gath-Geva Clustering:** The Gath-Geva is able to identify the desired covariant membership functions μ_j. It uses for initialization a subset $\mathcal{V}^{C_I}$ of the cluster centers $\mathcal{V}^C$ which were determined through the subtractive clustering. The Gath-Geva clustering is numerically instable and does need the amount of clusters in advance, which is compensated through the subtractive clustering.

   Input: Training data $\mathcal{V}_i^{tr\cup\overline{c}_i}$ for the training of module $\mathbf{M}_i$ and a set of cluster centers $\mathcal{V}_i^{C_I}$ and an upper bound n_C.
   Output: Membership functions μ_j of mapping function $\mathbf{S}_i$ of module $\mathbf{M}_i$ as parameters Σ_j^{-1} and $\overrightarrow{\mathbf{m}}_j$. Also the number of rules $m = n_{C_I}$ is passed.

5. **Linear Regression:** A least squares method is used to determine the parameters $a_{1j}, .., a_{(n+1)j}$ of the linear functional consequence f_j of the mapping function $\mathbf{S}_i$. According to the training data, the designated output and the membership functions determined through the Gath-Geva clustering a linear equation is set up.

   Input: Training data $\mathcal{V}_i^{tr\cup\overline{c}_i}$ for the training of module $\mathbf{M}_i$. Membership functions μ_j of mapping function $\mathbf{S}_i$ of module $\mathbf{M}_i$.
   Output: Linear functional consequences f_j as parameters $a_{1j}, .., a_{(n+1)j}$ of mapping function $\mathbf{S}_i$ of module $\mathbf{M}_i$.

6. **Fuzzy Inference System (FIS) Construction and Output Determination:** After the Gath-Geva clustering and the linear regression all components for constructing a TSK-FIS are available. When the FIS $\mathbf{S}_i$ is constructed, the output on the training $\mathcal{V}_i^{tr\cup\overline{c}_i}$ and check data $\mathcal{V}_i^{ck\cup\overline{c}_i}$ can be determined. The output $\mathbf{S}_i(\mathcal{V}^{tr\cup\overline{c}_i})$ is fed back to step 1 as additional dimension to the training data. This is only done if the inner loop has terminated and the genetic algorithm has found the optimal subset of cluster centers $\mathcal{V}^{C_I}$. Also, the output is only calculated on the FIS that results from the optimal subset.

   Feedback: Output of the best mapping function $\mathbf{S}_i(\mathcal{V}^{tr\cup\overline{c}_i})$ is fed back after termination of inner loop.
   Input: Training data $\mathcal{V}_i^{tr\cup\overline{c}_i}$, check data $\mathcal{V}^{ck\cup\overline{c}_i}$, linear functional consequences f_j as parameters $a_{1j}, .., a_{(n+1)j}$ and membership functions μ_j as parameters Σ_j^{-1} and $\overrightarrow{\mathbf{m}}_j$ of mapping function $\mathbf{S}_i$ of module $\mathbf{M}_i$.
   Output: Results of the mapping function on the training data $\mathbf{S}_i(\mathcal{V}^{tr\cup\overline{c}_i})$ and check data $\mathbf{S}_i(\mathcal{V}^{ck\cup\overline{c}_i})$.

7. **Fitness Calculation (MQE):** The genetic algorithm needs a fitness value for each individual to optimize upon. This fitness value is the inversion of the mean quadratic error, the added quadratic distance of output to designated over the whole training data. The passing of the fitness value causes the inner loop. The loop is executed for every individual of every generation and epoch.

   Inner Loop: The fitness value is provided to the genetic algorithm.
   Input: Training data $\mathcal{V}_i^{tr\cup\overline{c}_i}$, mapping function output $\mathbf{S}_i(\mathcal{V}_i^{tr\cup\overline{c}_i})$ and designated output $\overrightarrow{\mathbf{u}}_i$.
   Output: Fitness values $\mathcal{F}^{(l)}(\mathcal{V}_i^{tr})$ of individual l.

8. **Stop Criterion:** There are three criteria qualifying for determination when the algorithm should stop. One is the mean quadratic error of the check or training data, which was already used in the inner loop. This produces activity recognition modules, which are highly expressive according to the reliability measure, but lower recognition rates. The second one is the percentage of correct classifications on the check and training data. This optimization criteria would lead to a mostly correct classifier, but the expressiveness of the reliability value would be lower. Since the reliability value is used for filtering, a high expressiveness would indirectly lead to a high classification rate. The third criteria would just be a counter, counting the number of outer loops. If the previously specified number of iterations is reached, the algorithm is stopped. This criteria requires the saving of all iterations results, so the expert can make a choice about the iteration whose result is used. The different approaches are evaluated and discussed later in this thesis in subsection 4.3.

   Outer Loop: If stop criteria is not met goto step one of the algorithm, otherwise algorithm stops.
   Input: Mapping function output of training $\mathbf{S}_i(\mathcal{V}_i^{tr\cup\overline{c}_i})$, check data $\mathbf{S}_i(\mathcal{V}_i^{ck\cup\overline{c}_i})$ and designated output $\overrightarrow{\mathbf{u}}_i$.
   Output: If algorithm stops, the output is the final RFIS mapping function $\mathbf{S}_i$, all mapping functions of other iterations and the performance values produced through the algorithm.

The algorithm presented here provides the recurrent mapping functions for a modular classifier. The recurrence offers **robustness** and enables the determination of a reliability measure. Also, the algorithm considers the identification of a special complementary class, which is used to determine a module switch. A module which is trained on one dataset might be personalized on another dataset, generalized or just be changed. How this works is described in the next subsection.

3.3.7 Bit-Vector Search

If user or environmental changes occur (e.g. different user, clothes, environment, physical or mental state) the classifier modules need to be adaptive of these changes. This adaptation is done via bit-vector masking, which was described in subsection 3.1.6. Here the identification of such a bit-vector is described.

Data for Adaptation The data the bit-vector is identified upon has to fulfill some conditions. First, if only for a subset of the overall modules a bit-vector is searched, the newly provided data does not include a sufficient amount of data to represent the complementary class. In this case the original training data for the complementary class can be added to the new data the bit-vector is identified upon. Second, if a module needs to be adapted all its classes need to be represented through the new data. Otherwise, the classification of the classes not present in the new dataset would be worsened through the adaption, since the classifiers capabilities get shifted towards the present ones. If new data is not present for all classes, the dataset has again to be filled up with the original training data. Third, the data has to be specific to the changes the respective modules need to be adapted on, otherwise the adaptation through the bit-masking would not be effective.

Fitness Value Again, two criteria are qualifying for fitness values, the percentage of correct classifications or the mean quadratic error of the newly acquired data. As for the selection of best initial cluster centers, the choice is the mean quadratic error, since it not only delivers good classification results, but also an expressive reliability measure. The fitness value function (eqn. 44) would therefore be the same as in subsection 3.3.5. In the evaluation in subsection 5.1 the two metrics are compared.

Individuals Genome The individuals genome is the bit vector, which is used for masking. The fitness value for the respective individual is calculated according to the mean quadratic error or the percentage of correct classifications upon the classification with the masked RFIS of the new data.

Genetic Algorithm Also, the genetic algorithm used to hierarchically optimize the bit-vector is the same as used for the identification of the best subset of cluster centers, which was described in subsection 3.3.5. Here it is used to search for a bit-vector which is used to mask the RFIS mapping function in classification modules. The genetic algorithm searches for a bit-vector, which masks the mapping function so that a minimal error on the data on which the module should be adapted on is obtained.

The complexity of this search is much higher than for the best initial cluster centers for the Gath-Geva clustering, since each rule is not only represented by a bit, but each rules dimension. Therefore, the search space is 2^{b_i} with $b_i := n \cdot m_i$ of input dimensionality n and number of rules m_i for module $\mathbf{M}_i$. For each individuals bit-vector the fitness value has to be calculated over the whole set of data the module needs to be adapted on.

More details on the bit-vector search are described in the respective subsections of the challenges **flexibility** and **extensibility**.

3.4 Tools Supporting the Modular Classifier Architecture and Implementations

In the previous subsections the general sensor data processing, the modularity of the activity recognition and the machine learning algorithm to train the modules mapping function were defined. Since the machine learning needs training data, data needs to be collected on the device. Also, the data needs to be annotated, so the training has a goal the mapping function should be fitted onto. The steps of data collection, annotation and classifier training need to constantly be repeated, since various users, conditions and phones are supported. Therefore tools have been developed, which are easy to use and help speeding up the training and deploying of new classifiers. Furthermore, the peer group of the novel modular activity recognition is the common user not only the researcher, therefor wide spread mobile phone platforms are supported due to implementation.

First, the implementations of the modular activity recognition for different mobile phone architectures are described. The phones of choice are the OpenMoko Neo Freerunner for research, the various phones running Android and the Apple iPhones. A basic classifier code structure was implemented, which runs efficiently and is native compilable for all architectures.

Second, the Classifier Specification Format (CSF) is defined, which is used to transfer the trained classifier modules from the server, where the training algorithm was running, to the respective mobile phone. The format needs to be in a script language where there are open source parsers for many programming languages. Also, the files in the CSF, used to transmit the parameters of the classifier modules to the devices, need to be small in size, so the limited network bandwidth is not stressed too much. Another issue is the limited storage space on the mobile phone, which might need to store a lot classifier parameter files.

Third, the implementations of the training algorithm are described. For rapid prototyping and research, a first implantation was done on Mathworks Matlab. Since Matlab is not an open source software and demands a license to use it, a second implementation was done, which can openly be distributed and used on servers. This second implementation was done in Python including Numpy packages.

Fourth, the Context Annotator Tool (CAT) is introduced. This tool is used to annotate non annotated sensor data according to markers in a simultaneous recorded audio stream. With the CAT already annotated data can also be loaded and modified. The CAT implements an easy to use graphical point-and-click method to place annotations. It also includes various ways for importing and exporting sensor data.

Fifth, the three implementations of the Data Collector Tool (DCT) are explained, which are used to collect data on the mobile phones. There are two ways of collecting data: one is to record sensor and audio data, so the annotation can be done in the CAT; the other one is to collect data and directly annotate it on the device. For on device annotation the DCT provides a counter so the user can prepare for the respective activity. Also, signal noises indicate the start and end of annotation.

Last, the Annotation Package Format (APF) is introduced, in which the annotated data of various sensors is stored. The DCT directly stores the data in the APF, where the CAT and the Python training algorithm can load this format. Also, the CAT saves the data in this APF when the annotation is done with it or annotations are modified.

3.4.1 Implementations for Different Phones

The novel modular activity recognition presented in this thesis is not only for research purposes, but should be available to the common user. Therefore the implementation of the classifier functions was done on several phones. For research purposes and to have an upper limit the OpenMoko Freerunner phone was used, which runs a Debian distribution. Most of the common users can be reached through an implementation for the Android operating system, which is supported by many mobile phones from various vendors. Another widely spread mobile phone is the iPhone using the iOS, which is also supported through an implementation of the modular activity recognition.

Design Aspects Where the GUIs need to use the graphical elements supported by the respective operating system, the implementation of the modular activity recognition needs to be as efficient as possible. The classification has to be calculated many times per second. Although through the modularity the calculation effort was limited to a fraction compared to a monolithic architecture (see subsec. 4.4), the processing still needs a large portion of the processor load if the classifier is running e.g. in Java (Android). Therefore a native implementation of the recognition was preferred over simple implementations offered through IDEs of the respective operating systems. Also, the code in the chosen language C can be cross-compiled for any ARM based processor architecture, where

e.g. Java is not supported on iPhones. This saves development time, since not every architecture needs its own implementation. Furthermore, having various implementations for many operating systems would lead to a lot of problems mastering bugs in the code. If only one cross-compilable code base has to be maintained, the debugging and further development is much more manageable.

Another design aspect is about what parts of the modular activity recognition should be in the native C or in the respective supported programming language. Since the whole recognition for the OpenMoko Freerunner is done in C without any GUI, the whole processing queue exists divided in different libraries. When an architecture just supports the readout of the sensors through the API (e.g. Anroid) the native processing queue can be picked up at the feature extraction or the mapping function. This offers maximum flexibility and different strategies can be tested on different operating systems.

Architecture The architecture of the native modular activity recognition in C contains the following libraries which are listed alphabetically:

acceleration-replayer.c/.h Plugin to replay acceleration sensor data from a file.
Includes: replayer.h

acceleration.c/.h Reads the accelerometer sensor and passes the data.
Includes: config.h, performance.h, plugin.h

audio.c/.h Reads the audio sensor and passes the data.
Includes: performance.h, plugin.h, kiss_fftr.h

battery.c/.h Reads the remaining battery level and passes the value.
Includes: -

broadcaster.c/.h Plugin to transmit recognized activities via UDP broadcaster.
Includes: -

classifier.c/.h Main file executed from command line or called from other instances to classify sensor data.
Includes: audio.h, rule.h, acceleration.h, acceleration-replayer.h, performance.h, broadcaster.h, classifier_set.h, battery.h

classifier_set.c/.h Parses the JSON files and passes the rule sets of the classifier modules.
Includes: rule.h, plugin.h

config.h Configuration file specifying parameters of audio and accelerometer sensor.
Includes: -

gen.sh Script to make the classifier
Includes: -

kiss_fftr.c/.h Feature extraction KISS FFT package.
Includes: -

performance.h Definition of performance measurements.
Includes: config.h

plugin.c/.h Plugin for command line sleep time scheduling of activity recognition.
Includes: rule.h, performance.h, scheduling.h, battery.h

replayer.c/.h Plugin to replay sensor data from files.
Includes: plugin.h

rule.c/.h Calculation of FIS mapping through rule evaluation.
Includes: vec.h, config.h

scheduling.c/.h Counts activities for scheduling plugin.
Includes: -

types.h Definition of data types for audio and accelerometer sensor.
Includes: config.h

vec.c/.h Definition of vector types and operators.
Includes: -

Graphical User Interfaces (GUI) The GUI differs from platform to platform. The OpenMoko implementation is just a command line classifier, which only uses the previously listed C files.

The GUI for Android platforms offers an operation concept which is more suited for the common user. The Eclipse IDE for Android offers a GUI designer, where control elements can be placed on a window with drag and drop methods. The elements and the parameters are interpreted as XML code. After the activity recognition application is started, the user is constantly informed about the currently recognized class through pictures visualizing the activity. Other windows are selected through the menu, which offer additional GUIs for data collection or specifying activity dependent phone profiles.

The GUI for the iOS is similar to the Android one, since it also uses pictures to display the currently recognized activity. Other parts are quiet different because the Apple GUI elements differ from the Android//Java elements. The iPhone App only has two different windows, the main screen with a data *Collect* button and an *Online* button and the options menu to specify a FTP server address, at which the sensor data is sent after collection. The *Online* button followed by a double tab on *Start* is used when the online activity recognition should be activated. With the *Collect* button the sensor data collection can be triggered, which is explained in detail in subsection 3.4.5.

Currently both implementations for iPhone and Android are redesigned, since experience with the operation of the applications revealed serious problems. This experience was gained during a field trial with the iPhone application and 30 test users [68]. Also, due to the service (sec. 6) the activity recognition becomes usable for common users. Due to this the GUI must become much simpler, so the user immediately can operate the application. The evaluation of the redesign is part of future work.

Implementations The implementation of the modular activity recognition, which only uses C code is the one for the OpenMoko Freerunner running a Debian distribution. Only the Data Collector Tool (DCT) for the OpenMoko, which is described in subsection 3.4.5, is implemented in Python. The activity classifier itself is accessible through the command line with the following command and parameters:

```
Usage: classifier CJSON [-hcbs] [-f FORMAT] [--help] [--cpu-runtime]
                    [-udp-broadcast] [--format FORMAT] (PLUGIN CLASSIFIER)*

   -h,--help Show this help
   -c,--cpu-runtime Print runtime statistics upon exit
   -b,--udp-broadcast Broadcast classification results via UDP
   -f,--format Output the classification result using a format string
   -s,--scheduling  Use scheduling algorithm to conserve energy

  FORMAT is a string that can contain the following letters:
   's' The current time in seconds
   'u' The microseconds of the current time
   'c' The classification result (a string)
   'r' The raw value of the classification
   'g' The expected classification result (ground truth)
   'b' The current battery level (in percent)

  PLUGIN is a string seperated by ':' describing a plugin and its options.
   Available plugins: acceleration
    Replay acceleration sensor data from FILE: replay:acceleration:FILE

  CJSON is a json file describing a classifier.
```

The classification output can be transmitted via UDP (*-b, –udp-broadcast*), printed in a console according to formatting parameters (*-f, –format*) or written in a text file (> *file.txt*). Various measurements of CPU usage (*-c,–cpu-runtime*), battery consumption (*'b'*) and time stamps (*'s'* and *'u'*) are offered. Also, a sleep time scheduling of the activity recognition can be activated (*-s, –scheduling*). The activity recognition can be cross compiled for any ARM architecture and is compiled for OpenMokos with the *gen.sh* script.

The C code of the classifier is called from the Java code in the implementation of the activity recognition on Android phones. The sensor readout, feature extraction, data transmission and classification result output is all done in Java, since the API for Android is pretty strong and easy to handle. The classifier in C is used to have a high performance recognition, which is equal on every used operating system. Android applications are developed in Eclipse through a plugin and the Standard Development Kit (SDK) for Android. The C code is natively compiled through the usage of the Native Development Kit (NDK).

In the implementation of the application on iPhones, the C code is just embedded in the Objective-C code natively supported by the iOS. The compiler has to be switched to a dual mode, where it can compile both, the C and the Objective-C code. The C activity recognition files are added to the normal project. Reading the sensors is again done via the API, but for the feature extraction the C libraries are used.

Both implementations for Android and iOS do not support the sleep time scheduling yet. Since the sleep time scheduling needs access to the threading mechanisms, for which it is currently not known whether the C libraries can be used, a implementation in the respective supported programming language would be preferred. Also, the modular activity recognition uses only a small fraction of the CPU time, since these phones include a FPU as well. Therefore, the energy consumption of the activity recognition is much lower than on an OpenMoko phone. Evaluation results in this manner are presented in subsection 5.4.

3.4.2 Classifier Specification Format (CSF)

Since the classifier modules are trained offline the parameters of the activity recognition systems need to somehow be transferred to the mobile phone. On the device the native classifier needs to parse the parameter files and include them into a running activity recognition. The question is now in what kind of format these parameters should be transferred to. A proprietary format could be constructed, but this would need a new implementation of a parser. There are also several standard formats qualifying as parameter files for the parameters of the classifier modules, e.g. XML. Since the calculation code of the classifiers is already coded in *C* the parameter format should be supported through the programming language. Also, the resources on the mobile phone and the network connection, where the files are transmitted over, should be most efficiently be used. Therefore, the JavaScript Object Notation (*JSON*) format was chosen, which is more efficient than XML and parsers for *C* are existing.

Format Layout The Classifier Specification Format (CSF) in *JSON* is exemplary displayed in figure 19 and described in the following:

- **Type:** The *type* specifies the sensor, which is processed through the classifier modules in this *JSON* file. This option is for future use, when other sensors despite of the accelerometer are also processed.

- **Dimension:** For input of the classifier modules, the number of *dimensions* needs to be specified. This dimensionality concerns the feature vector and the mapping functions of the classifier modules. The *dimension* directly corresponds to the number n in subsection 3.1.

- **Semantics:** Through the entry *semantics* the composition of the input feature vector is specified, what kind of extraction methods are used on which dimension of the used sensor. Also, the position of the respective feature and sensor dimension in the input vector is specified. The value "*last*" specifies the input of the last mapping result and therefore denotes the recurrent edge of the RFIS.

```
{
 "type": "movement",
 "dimension": 7,
 "semantics": ["mean_x1", "mean_y1", "mean_z1", "var_x1", "var_y1", "var_z1", "last"],
 "classifier": [
  {
   "rules": [
    {
     "consequence": [0.00038377003539143947,... ],
     "sigma": [7.5483206460979147e-08,... ],
     "mean": [1140.7277960526317,... ]
    },
    {
        ...
    }
   ],
   "name": "M1",
   "mapping": [
    {
     "class": "PSI",
     "value": 1.0
    },
    ...
   ]
  },
  {
   ...
  },
        ...
 ]
}
```

Figure 19: Classifier modules in *JSON* format.

- **Classifier:** The entry *classifier* encapsulates the whole parameters for all the classifier modules. Each of the classifier modules compose of a *name*, a *rules* and a *mapping* part.

  - **Rules:** Each of the entry *rules* encapsulates the parameters of a mapping function for one module. The *rules* (see eqn. 11) compose of antecedence and a consequence part. The *consequence* is a linear function and the antecedence is a Gauss function consisting of covariance matrix and mean vector. All parameters are in a high precision fixed point notation.

    * **Consequence:** The *consequence* of each rule of the RFIS (see eqn. 13) mapping function is specified through linear parameters. These parameters are listed under the entry *consequence*, where the first parameter of the entry corresponds to the linear weight a_{1j} of the rule j.

    * **Sigma:** With *sigma* the inverted covariance matrix Σ_j^{-1} of antecedent Gaussian membership function μ_j (see eqn. 10) is denoted. Here the inverted matrix is used so it has not to be inverted on the device, which would unnecessarily need additional calculation. In the notation of *sigma* the rows of the matrix are written consecutively in one vector.

    * **Mean:** The *mean* states the mean vector $\vec{\mathbf{m}}_j$ of the antecedent Gaussian membership function μ_j (see eqn. 10). The first value in the entry is the first value of the vector.

  - **Name:** The *name* is a string uniquely identifying the module whose parameters were listed before. The naming should include the original number k of the module specified in the training.

  - **Mapping:** The *mapping* encapsulates the module class triples c_{kl} (see eqn. 24), with internal numerical class identifier l of module k and the global char triple 'XYZ' uniquely identifying the activity.

* **Class:** The char triple 'XYZ' uniquely identifying the activity is denoted in the entry *class*. This triple along with the reliability value is the output of the module if the classification was done on this class.

* **Value:** The numerical class identifier stated in the *value* entry is only used internally, since this class number can be repeated in other modules.

The files describing the classifier modules in the APF can be transferred as a whole or just partially, so new modules can be added to preexisting ones or modules can be exchanged. Also, the bit vectors can be modified in JSON files, so modules can be personalized, generalized or adapted so they can recognize transitions onto new modules. This feature is not implemented so far.

3.4.3 Training Algorithm Implementations

In subsection 3.3 the algorithm was presented to train a recurrent modular classifier structure. This algorithm needs implementations to train classifier modules in a real setting. There are two implementations, one in Matlab and the other one in Python.

Matlab Implementation During the research of the training algorithm, a method was needed to rapidly develop and test new variations of the algorithm. Also, various statistical and graphical methods were needed to evaluate and plot the results of the training. Since Matlab offers not only various plotting and statistical analysis tools, but also implementations of the subtractive and the Gath-Geva clustering, it was the obvious choice for rapid prototyping. The Gath-Geva clustering was not implemented by Mathworks, but can be found in a clustering toolbox for Matlab provided by Kenesei, Balasko and Abonyi [84].

The biggest disadvantage of a Matlab based implementation is, that the training algorithm is only accessible to users with a Matlab license. Since the tools, algorithms and the activity recognition itself should be available to the mass-market, this is unacceptable. Also, the tools and algorithms should be included in a service (see sec. 6) which supports the activity recognition on the user's phone. This especially rises the need to use an open source implementation where no conflicts with copyrights could occur.

Python Implementation The open source implementation of the training algorithm was therefore written in Python, which offers through SciPy a large toolkit for scientific data processing. Also, the NumPy packages help to produce very efficient code, which is marginally slower than Matlab implementation or a C/C++ realization of the training algorithm would be [1].

The training algorithm composes of the following Python scripts in alphabetical order:

command_line.py Script to use the training algorithm on command line. Is especially needed in the realization of the service supporting the modular activity recognition.

evolve.py Genetic algorithm (see subsec. 3.3.5) to determine the best subset of initial cluster centers and to identify bit-vectors.

GathGeva.py Gath-Geva clustering (see subsec. 3.3.3) used to determine the Gaussian membership functions.

gui.py Starts the Graphical User Interface (GUI) which is used to execute the training algorithm.

Regression.py Least squares regression for determining the linear functional consequences (see subsec. 3.3.4).

SubtractiveClust.py Subtractive clustering (see subsec. 3.3.2) for initialization of the Gath-Geva clustering.

Training.py Overall training algorithm (see subsec. 3.3.6), that connects the clusterings and the linear regression to identify a modular and recurrent activity classifier.

The training algorithm can be started through the script *command_line.py* with the following parameters:

```
Usage: python command_line.py data.tar config.ini
                CLASSIFIER1:CLASS1,CLASS2,CLASS3#CLASSIFIER2:CLASS3,CLASS4
                   classifiers.json

    data.tar          Specifies the file holding the annotated training/check data
    config.ini        Parameter file for the training algorithm.
    classifiers.json  Output file for the resulting classifier modules in JSON

    CLASSIFIER        All classes after the : are classified by this module
    CLASS             Classes in the triple char format separated by ,
    #                 Separates the different classifier modules
```

Graphical User Inteface (GUI) The GUI to handle the training of the classifier modules is based on the GTK+ library. After the GUI is started, the first window (fig. 20, left) requests the specification of a file holding the annotated training and check data. The data has to be in the Annotation Package Format (APF) described in subsection 3.4.6.

The user can proceed with the *Forward* button to get to the parameters window (fig. 20, middle). The options

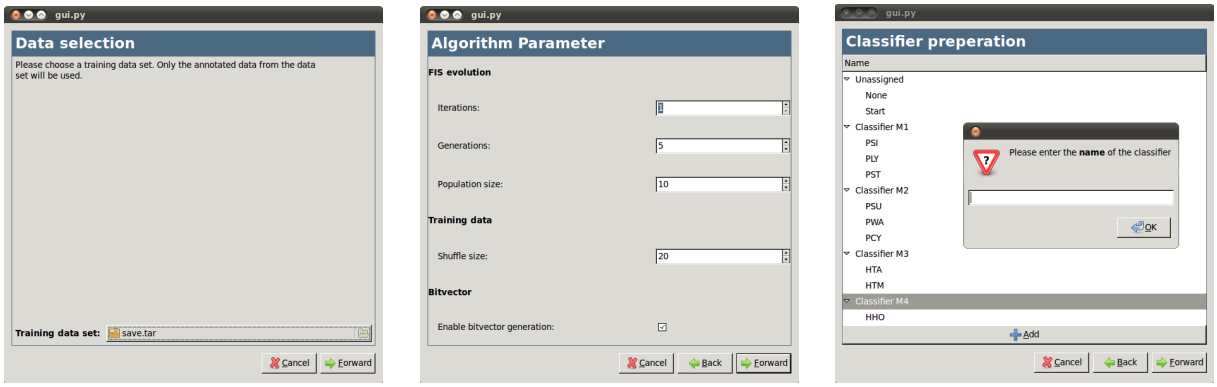


Figure 20: GUI of context trainer implementation in python with data load (left), parameter set (middle) and module assignment (right) windows.

in the parameter window are the following:

- **FIS evolution:** Here the parameters for the genetic algorithm selection of the subset of initial cluster centers (inner loop) from the subtractive clustering for the Gath-Geva clustering can be set. Also, the number of iterations (outer loop) for the training of the recurrence of the FIS can be set.
  - **Iterations:** Integer number of iterations of the outer loop of the training algorithm to train the Recurrent Fuzzy Inference System (RFIS).
  - **Generations:** Integer number of generations to evolve with the genetic algorithm to determine the best set of initial cluster centers.
  - **Population size:** Integer number of individuals in the population of the genetic algorithm of the inner loop.

- **Training data:** Since in the training of the RFIS the most false classifications are made in between the activities, the training and check data is shuffled in slices of a certain width.
  - **Shuffle size:** Here the width of the slices to be shuffled can be set as an integer number.
- **Bitvector:** After the training of the classifier modules a bit vector can be determined, which improves the recognition rate further.
  - **Enable bitvector generation:** If a bit vector on the check data should be determined is enabled through this option.

In the next window (fig. 20, right), the user can determine how many modules should be trained and what module should classify which activity of the previously loaded data set. New modules can be added through the *Add* button, which triggers a new window, where the name of the module can be set. Via drag and drop, the activities of the previously loaded data set can be assigned to the respective modules. The activities are uniquely identified through a char triple described in subsection 3.1.4.

The training of the classifier modules starts after another click on the *Forward* button. The intermediate results of the training are only shown when the GUI was started from a console. Depending on the previously set parameters, the length of the data set and the amount of activity classes, the training lasts from several minutes to several hours.

After the training has stopped, a change into the next window offers the choice to save the classifier modules. The module parameters are saved in the Classifier Specification Format (CSF) (see subsec. 3.4.2) in a JSON file. The file name and the destination has to be entered by the user. This file can than be run on any of the three platforms supported so far.

3.4.4 Context Annotator Tool (CAT)

While dealing with collection of training data for machine learning algorithms recurring problems occurred. If the data is annotated offline, the audio stream can be used to set markers. The problem here is to synchronize and visualize the accelerometer and the audio stream. Having the streams synchronized and visualized, the data has to be annotated. Here a point and click method makes the annotation much simpler compared to other methods. This offline method offers the opportunity to collect large naturally occurring datasets without loosing the ability of an afterwards reconstruction of the activities which happened.

If the annotation is done online, the Context Annotator Tool (CAT) offers the opportunity to adjust, delete, add and check data or annotations. The CAT additionally implements functions to import and to export from and into different data formats. This enables the usage of different training algorithms or implementations, such as the various tools in Mathworks Matlab or other open source toolkits.

A similar tool for data annotation, which also supports video streams, was developed in parallel to the CAT and presented in [17, 18], but with different focus. The CAT is only means to an end, to support the novel modular activity recognition and not to support data annotation in general. It could be extended to also generally support the data annotation, but this is out of focus of this work.

Purpose To generate data for the training algorithm, the collected accelerometer sensor data has to be annotated with the activity in which it was collected. In order to associate sensor data with context information, it is necessary to...

1. read the sensor data from audio and acceleration sources,
2. display the sensor data as different streams,
3. allow the user to annotate the data with context information,
4. export the annotated data into a format suitable for the training algorithm,
5. save and load the data, so it can be modified.

The Context Annotator Tool was designed to put all these mechanisms into one easy-to-use tool.

Data Acquisition There are three ways (see fig. 21) to support data for the training algorithm of which two involve the CAT. The first way is to directly annotate the data on the mobile phone using the Data Collector Tool

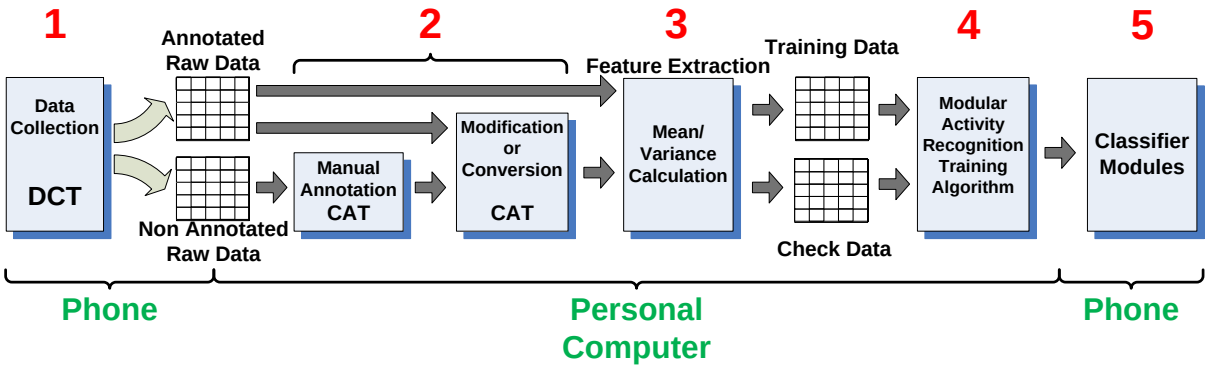


Figure 21: Various methods of data acquisition for training algorithm.

(DCT) and directly passing it to the feature extraction step without using the CAT. This procedure is described along with the DCT in subsection 3.4.5. Second, the annotated data from the DCT is opened in the CAT, so it can be modified or converted. The third way is the primary purpose of the CAT, the annotation of non annotated sensor data. Therefore, the CAT implements a *Load Sources* method (see fig 22), where audio and accelerometer data streams can be loaded. For the audio data a offset needs to be entered, which aligns the visualization of the stream

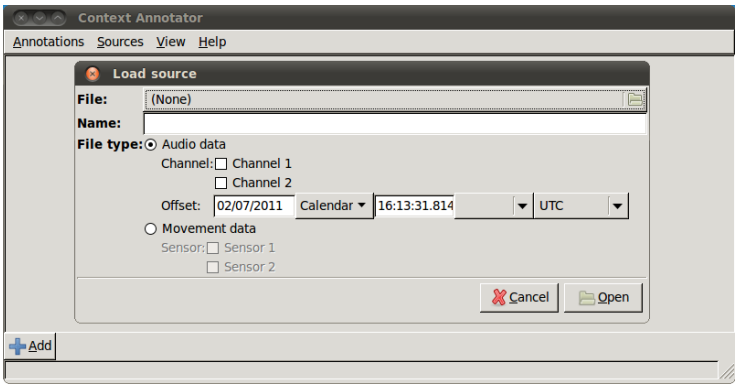


Figure 22: *Load Sources* menu of CAT.

with the accelerometer data. Also, the number of channels need to be specified, since the *OpenMoko Freerunner* has two microphones. The import of the acceleration data only needs the specification of the amount of acceleration sensors, one or two. The design of the CAT generally would imply, that it supports more sensors than just audio and acceleration, but since this thesis is about activity recognition with accelerometers, these two are the ones supported so far.

Design aspects Most sensor data can be visualized as a two dimensional graph with a time and value axis. To be able to distinguish between different sensors, each sensor data should be displayed separately, but it should nonetheless be possible to see which sensor data happened at the same time. To achieve this, a horizontal layout, with the sensor data aligned to a global time axis, is a good choice.

Good visualization of the annotations can be achieved by overlaying the graph with colored boxes representing the activities. To ease the process of associating a activity to an audio sample, playback of audio samples was integrated by allowing the user to select a time slot and playing the contained audio data.

To be able to load and store annotated data, a special data format was designed, which is described in subsection 3.4.6. It not only allows the reloading of data in the CAT, but is also used in the DCT for storing online annotations and in the implementation of the training algorithm to easily import the data.

Implementation The CAT (see example in fig. 23) is written in Python to provide good extensibility, platform-independence and speed of development. For the data visualization the *Matplotlib* [77] toolkit was used, which provides Matlab-like plotting capabilities. The GUI is implemented on top of the *GTK+* toolkit to allow the tool to run on as many platforms as possible. Audio loading, processing and playback is done using the *GStreamer* framework [3], which frees the implementation from worrying about things like audio codecs and output.

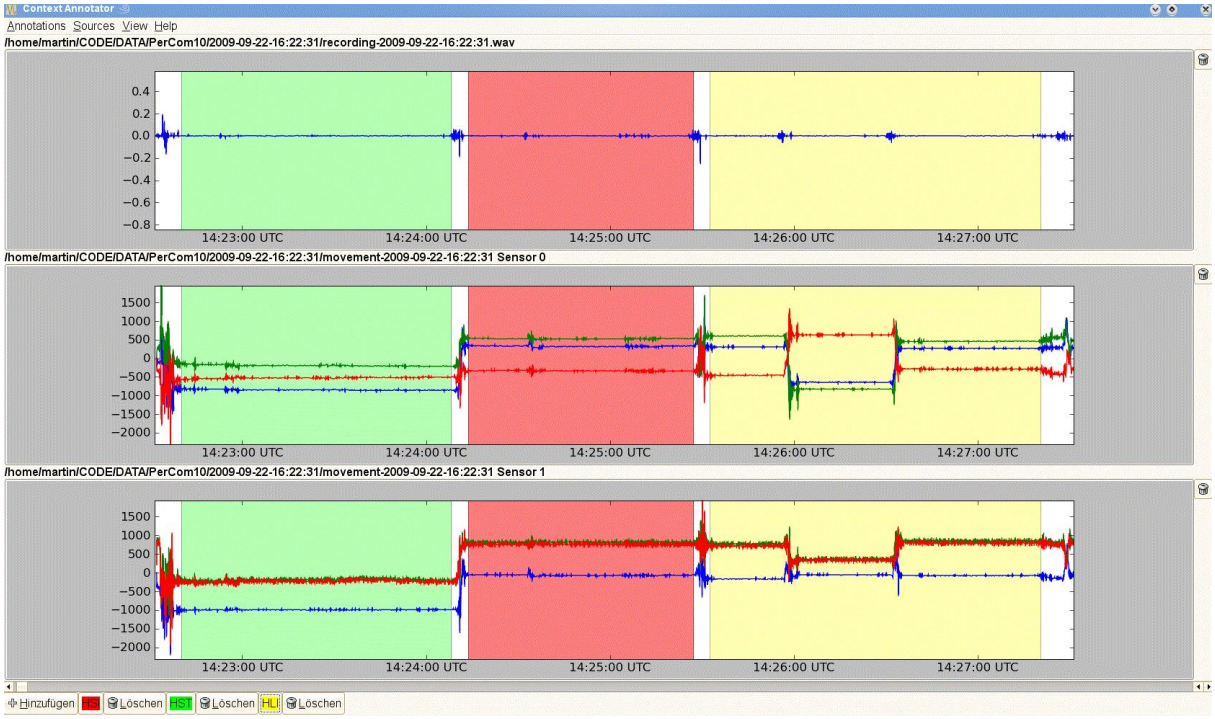


Figure 23: Screenshot of Context Annotator Tool (CAT) with audio data (upper stream) and data of two accelerometer sensors (middle and lower stream) from an *OpenMoko Freerunner* phone.

The central data structure of the tool is the *annotations* structure. It keeps track of all activity annotations (start-/end-time and name). Other components can register as listeners if they want to be informed if an annotation is added, deleted or updated. The internal time stamp formats used within the *Matplotlib* for the annotations where also used for storing and loading annotated datasets. More on the Annotation Package Format (APF) and the time stamps of the annotations is described in subsection 3.4.6.

Each sensor data display is represented by a *display* component. It renders the sensor data using *Matplotlib* and reacts to user interaction, such as scrolling, adding new annotations, etc. The sensor data is managed by the *source* component (fig 22). It is responsible for loading sensor data from various formats (audio file, accelerator log file, etc.).

The Context Annotator Tool (CAT) offers a strong support for acquiring data on which the novel modular activity recognition is trained on. It enables the offline annotation of collected data according to audio markers, can be used for editing data which was annotated on the device and supports several methods for importing and

exporting data. To provide data from the mobile phone a Data Collector Tool (DCT) was developed, which is described in the following.

3.4.5 Data Collector Tool (DCT)

To collect data for the training algorithm, a Data Collector Tool (CAT) was implemented for different mobile phones. The DCT needs to offer the ability to collect data and directly annotate it on the device. Therefore selection menus offer a set of activities to pick, which triggers a countdown in which the user has to prepare for the activity, e.g. putting the phone in one's pants pocket. After the countdown a signal noise is occurring indicating the start of the activity. When a certain time has passed - the timespan depends on the amount of data needed for the training algorithm - a second signal indicates the end of the recording for that activity. The data is saved in the Annotation Package Format (APF), which is described in subsection 3.4.6.

If the annotation is not done online, the DCT offers the opportunity to also collect audio data, so markers for the offline annotation can be recorded. The non annotated data also gets saved in the APF.

Implementations Since three different phone operating systems are supported (Debian on the OpenMoko Freerunner, Android and iOS on iPhones) the implementations vary, especially in the appearance of the GUI. All three implementations are displayed in figure 24.

Figure 24: Data collector implementations for OpenMoko Freerunner (left), Android (middle) and iPhone (right)

The most basic one is the GUI implementation for the OpenMoko Freerunner phone, since this is not an everyday mobile device and mostly only used for research purposes. This phone was, in this case, also used for rapid prototyping and to provide an upper limit of what is possible, since the phone has two accelerometers and offers sampling rates which are not provided though other APIs. The GUI was implemented in *Python*, so it can share the libraries of the Annotation Package Format (APF) with the CAT. This implementation does not include a countdown until the respective activity should be performed and the annotation does not stop after a certain time period. This limitation can be coped with through the modification of the annotations in the CAT.

The second GUI implementation is for the iOS of the iPhone family (fig. 24, right). Here a *picker* provides a selection of different standard activities. After the selection of an activity and the following countdown, an audio signal occurs which indicates the start of the annotation. A second signal indicates the end of the annotation after which a new activity can be selected from the *picker*. The selection of activities can in a separate menu be modified. New activities can be added, old ones be edited or deleted. The DCT implementation for iPhones also includes the capability of collecting audio data, which can be used for offline annotation. The GUI for collecting data is implemented in *ObjC*.

A wide spread operating system for mobile phones is the Android system. The main supported programming language for Android is *Java*. Android is supported through the IDE *Eclipse* and various API methods in the Android SDK. The DCT implementation (fig. 24, middle) also includes a drop box where the respective activity can be selected, followed by a countdown and audio signals indicating start and end of annotation. Since most mobile phones running Android support various sensors despite acceleration and audio, the GUI includes the option for recording them, too. Nevertheless, the modular activity recognition does not include these sensors so far.

Currently the implementations of the activity recognition implementations for Android and iOS are redesigned, which includes the DCT. The operation of the DCT needs to be optimized for the usage within a service (sec. 6). Also the common user should immediately and intuitively be able to operate the DCT. All of the DCT implementations on the different operating systems utilize the Annotation Package Format (APF) for data recording, which is defined in the following.

3.4.6 Annotation Package Format (APF)

Since various implementations of the DCT produce annotated sensor data, the CAT is used to visualize and modify the data, the data is stored in a database and the training algorithm finally utilizes this data, a data format is needed, which most flexibly supports all of these tools and algorithms. To not mix up data and annotations (which is creating the need for conversion tools) the two are handled in separate files. Also, the different sensors are recorded in separate files. This enables the recording of very different sensor types, as the currently supported audio and accelerometer sensors. Even for future use the APF is suitable, since this separation allows the recording and processing of many more sensors.

Format Design An annotation package is a file that combines sensor data from multiple sources with annotation data. The APF is displayed exemplary in figure 25 and described in the following:

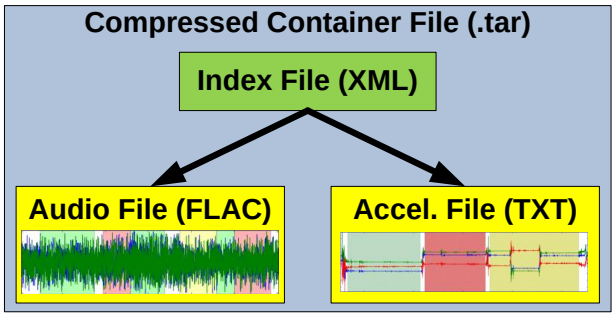


Figure 25: Annotation Package structure

- **Archive (tar):** The outer layer of the file is a *tarball*-archive. It wraps all the data files and the index file together. The tar format also supports compression, which is especially useful for all sensor files, since they are mostly text files and therefore extremely shrinkable. This is an important issue on storage size limited devices as mobile phones.

- **Index (XML):** The index file is coded in XML and lists all the different sensor files and the types of sensors. It also includes the annotations *begin* and *end* time if there are any. The index file is mandatory.

- **Sensor files (text):** The acceleration sensor measurements are written in plain text files. The files consist of many columns, where the first always lists the time stamps in the UNIX/POSIX format. In case of an three axis acceleration sensor, the following three columns are the measurements for x-, y- and z-axis. If there are more than one acceleration sensor, such as the OpenMoko Freerunner has, the different sensors measurements are listed in different files. Other sensors like temperature, magnetic field, proximity, or light could be

listed in such text files, too. Here the dimensionality decides on the amount of columns.

- **Sensor files (audio):** The recoding of the audio input is another issue, since the different operating systems support different audio codecs. The OpenMoko Freerunner with a Debian operating system supports nearly all formats. Since audio recordings in .wav format need a lot of storage, which is limited on mobile phones, the format of choice for this architecture was the .flac codec. This format enables compression without loss. The Android phones usually use the .3gp codec, which uses for audio the AMR or AAC format. The iOS for iPhones uses .caf (Core Audio) format.

Index File Each file in the APF needs an index file, which lists all the sensor data files and the annotations. An example of an index file is listed in figure 26. The index file is written in non standard XML code, since there is no need of an automatic parser (in contrast to the CSF). Here the more clearly readable structure of XML is preferred over the more efficient JSON format. The files including the index file are wrapped and packed in a *tarball*-archive anyway, which shrinks the content significantly.

```
<?xml version="1.0" ?>
<sensory-input>
  <annotations>
    <annotation end="730121.960304" id="None" start="730121.957995"/>
    <annotation end="730121.962649" id="PSI" start="730121.960304"/>
    <annotation end="730121.962789" id="None" start="730121.962649"/>
    <annotation end="730121.965386" id="PST" start="730121.962789"/>
    <annotation end="730121.965732" id="None" start="730121.965386"/>
    <annotation end="730121.968146" id="PLY" start="730121.965732"/>
    <annotation end="730121.968341" id="None" start="730121.968146"/>
    <annotation end="730121.970707" id="PWA" start="730121.968341"/>
    <annotation end="730121.971298" id="None" start="730121.970707"/>
    <annotation end="730121.973626" id="PSU" start="730121.971298"/>
    <annotation end="730121.974513" id="None" start="730121.973626"/>
    <annotation end="730121.976708" id="HHO" start="730121.974513"/>
    <annotation end="730121.978795" id="None" start="730121.976708"/>
    <annotation end="730121.981023" id="HTA" start="730121.978795"/>
    <annotation end="730121.981846" id="None" start="730121.981023"/>
    <annotation end="730121.984104" id="HTM" start="730121.981846"/>
    <annotation end="730122.009006" id="None" start="730121.984104"/>
    <annotation end="730122.010821" id="PCY" start="730122.009006"/>
  </annotations>
  <sources>
    <movement file="0.log" name="0" sensor="0"/>
    <movement file="1.log" name="1" sensor="1"/>
  </sources>
</sensory-input>
```

Figure 26: Index file example.

The index files consist of the following parts:

- **Header:** The header is the standard XML header, which includes the XML version number.

- **Sensory input:** The entry *sensory input* wraps the *annotations* and the *sources*. Therefore, the *sensory input* marks the beginning and the and of the index file.

- **Annotations:** The entry *annotations* encapsulate all annotations to the sensor files. Each *annotation* includes the *id*, *start* and *end* time. The time stamps are in the Matplotlib native format, so load and save of the data files through the CAT do not cause conversion imprecisions. The *id* can hold any string, but usually the activities unique char triple (see subsection 3.1.4) is used.

- **Sources:** In the entry *sources* all sensor data files are listed. The *source* entry consists of the *file* name, the *name* and the *sensor* id. The *sensor* id uniquely identifies the type of sensor measurements listed in the respective file. The ids 0 and 1 are reserved for the acceleration sensors. Identification number 2 denotes the audio sensor being source of the file.

The Annotation Package Format (APF) is used at the sensor data source the DCT to store annotated data, in the CAT to load and save annotated data and in the training algorithm to load training and check data. The APF is designed to contain various sensor data and the data annotation in one compound file. Also, the APF files should be imported and exported into and from a database, but this database is currently not implemented.

4 Challenges

Activity recognition on mobile phones with the use of the internal accelerometer sensors offers a lot of opportunities. Since many users have mobile phones with sensors, this type of platform enables common users to include artificial awareness into their daily lives without the need of acquiring additional hardware. Many applications can profit from activity recognition, traditional ones as well as new ones emerging from mobile web usage. Nevertheless, acquiring activity information from and with mobile phones is very challenging, especially when using acceleration sensors.

Five challenges (fig. 27) were identified for activity recognition with acceleration sensors on mobile phones: **flexibility**, **extensibility**, **robustness**, **resources** and **conditionality**. A novel modular activity recognition has been proposed to deliver suitable solutions for all of these challenges together. In this section the proposed system is analyzed according to these challenges, in order to indicate its capabilities.

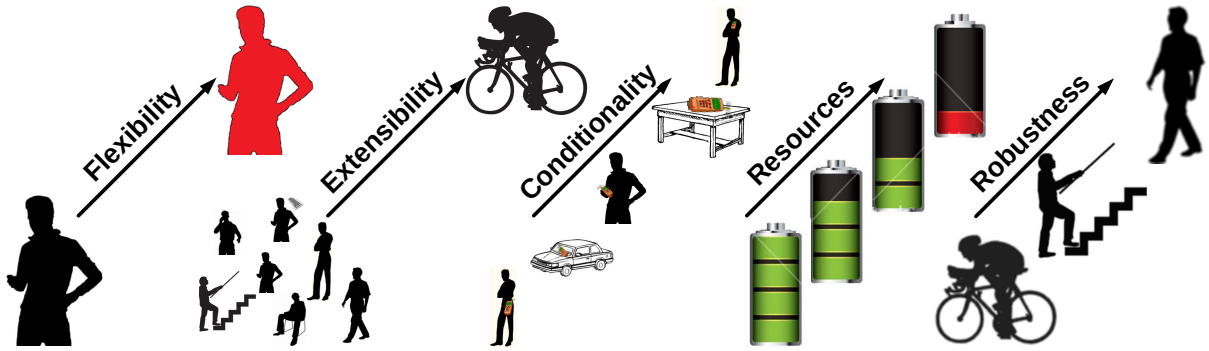


Figure 27: Challenges of activity recognition on mobile phones.

First, the **flexibility** of the novel modular activity recognition is analyzed. The approach delivers the opportunity to individually adapt or exchange certain parts (in this case modules) without the need to modify the whole system. This is especially of use with regard to the various situations and conditions the mobile phone can be used in. E.g. the user, the user's environment, physical or mental condition can change, which results in different senor data patterns and therefore affects the original system's performance. The methods adaptation and exchange to provide **flexibility** are described.

Second, the **extensibility** of the recognition system through new modules is analyzed. Most users perform a large set of standard activities, but each user is individual and e.g. engages in different sports, work and leisure time activities which are special to a certain group. A single system which supports all these user activities would need to recognize hundreds of activities. Such a system would be huge and probably have a very low recognition accuracy, even with the proposed modular architecture. Therefore, no single recognition system can support all these user demands. Instead, a base classifier needs to be **extensible** to the individual needs. If, whenever an **extension** takes place, the whole system would need to be re-identified, this would result in an enormous training effort. In the respective subsection therefore the problem of **extensibility** is described first and then the method of **extending** the modular activity recognition system is explained.

Third, the modular activity recognition is analyzed on **robustness**. Since the phone is not firmly attached to any position at the users body or in the environment, the acceleration data is noisy and the patterns vary according to the orientation of the phone. Therefore, the circumstance that the activity of a user does not change rapidly is utilized. In general, the probability that the current classification will be the same as the previous one is very high. Also, the reliability of each single classification varies strongly. Some sensor patterns are unequivocally assignable to a class, others are related to two or more classes. Since the novel modular activity recognition provides a reliability measure for every classification, the filtering upon this further improves the **robustness**. In the beginning of the subsection about the challenge **robustness**, various methods of deriving a reliability measure are listed and discussed. Then the **robust** classification is discussed, which is possible due to the recurrence of the classification process.

Fourth, the modular activity recognition is analyzed on its **resource** demands. One **resource** is the processor, the

other one the battery capacity. Since even mobile phone processors are becoming faster these days, the calculation effort for the classification becomes less important. Still, the processing does not only demand time, but also energy and this can not be consumed carelessly on mobile phones. Therefore, the reduction in processing time for the recognition also reduces the energy consumption. At first in the subsection of the challenge **resources**, the issues regarding processor load and energy consumption are discussed. A sleep-time scheduling method for the activity recognition is introduced, which further reduces the energy consumption.

Fifth, the capabilities of a modular activity recognition according to different **conditionalities** of the mobile phone and the user are analyzed. The phone can be carried in the different trouser pockets, in various pockets of the jacket, in a backpack or a purse. Also, the user can be sitting, standing or walking in a bus, train or car. The user can be left or right handed. All these conditions imply very different sensor patterns, which cannot be detected with only one classifier. Therefore, for each **condition** a separate module is used, which does not increase the complexity of the recognition, but improve the overall accuracy. At the beginning of the subsection about the challenge **conditionality** the capabilities of a modular activity recognition to react onto different **conditions** are discussed. Last, the possibilities to react onto different phone orientations with different modules is analyzed.

Finally, the **tradeoffs** between solutions for the different challenges that were identified throughout the investigations are analyzed. One **tradeoff** that was discovered is between the accuracy of the classification process and the so-called event detection rate. Due to the filtering upon the reliability measure, the overall accuracy could be improved, but filtering reduces the amount of classifications passed to the next instance. This reduction can result in the missing of whole activity event periods. Another **tradeoff** was identified between solutions for the challenge **robustness** and **resources**. The recurrence of the classification process improves the **robustness**, but due to the sleep-time scheduling, which reduces the **resource** consumption, the recurrence can have negative effects. The sleep phases result in leaps forward in time into a new activity, where the recurrent input dimension still includes knowledge about the old activity. All these **tradeoffs** are discussed and analyzed in detail.

4.1 Challenge 1 - Flexibility

In this subsection the proposed method of a modular activity recognition system for mobile phones is analyzed according to the first challenge **flexibility**. The **flexibility** of an activity recognition system was defined as the ability of the software to adapt to changed conditions. Changes of conditions can e.g. be caused due to different clothes, physical or mental states of the user or a different user at all. This adaptation should be done in a way, where only certain parts of the system need to be modified or exchanged, since only a certain group of activity classes could be affected. An example could be a physical condition called tremor, where the user is inadvertently shaking with her hands. This physical condition could be new to this user and would only affect activities where the mobile phone is held by the user. Other activities where the phone is carried in the pants pocket might not be affected through this condition.

First, in this subsection the opportunities, especially the exchange of modules of the activity recognition to cope with the challenge **flexibility** are discussed. The exchange of modules is explained and how modules can be modified, so they work with exchanged modules.

Last, the adaption of modules onto changed conditions is analyzed. The adaptation is done with a bit-vector masking, which does not destroy the original classification capabilities of the respective module. Also, the mask can at any time be removed, after which the original classifier is restored.

4.1.1 Exchange

An activity recognition used in real life has to face certain challenges, such as the **flexible** exchange of classifiers (fig. 28, left). With a modular structure the whole recognition system should not have to be exchanged, only the faulty module. Since the modules in the presented modular activity recognition do not only recognize the native activity classes, but also a so called complementary class, where on-recognition triggers a module reorganization in the dynamic queue, the exchange or adaption of one module solely is not just possible.

Figure 28: Examples of the exchange (left) and adaptation (right) of a classifier modules in a dynamic queue.

If a module in the dynamic queue gets exchanged, some of the other classifier modules might need to be adapted (see examples in fig. 29). This is because of the recognition of the complementary class. Since the modules which do not get substituted are not aware of the interchanged module, some of them falsely classify data, which should be classified by the substituted one. The modules, which falsely classify the data, do not recognize the complementary class, but a native class instead. Therefore they need to be modified in a way the complementary class is recognized instead of a native one. Experiments will show, that not all modules in the dynamic queue are problematic in this way and therefore do not need to be modified. Also, if the new module is not too different from the previous one, the queue could also work just fine without adaptations. Details about which or if modules need to be adjusted can be extracted from confusion matrices. Modules which are incompatible with the swapped module have mutually misclassified data with it. The adjustment of the modules is done via a bit-vector masking, which was described in subsection 3.1.6. To identify the bit-vector, the approach described in subsection 3.3.7 is used. Here the data used to train the swapped module is mixed into the training data of the complementary class of the modules to be adjusted. Therefore, the genetic algorithm searches for a bit vector, which modifies the respective modules in such a way the new data for the complementary class gets classified correctly. The masked modules then do not falsely classify the data for the swapped module, but transitions to this are made.

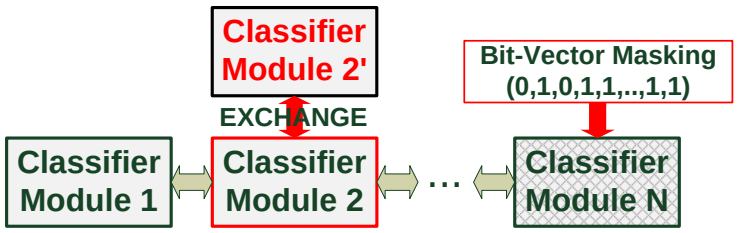


Figure 29: Example of exchange of a classifier module with additional bit-vector masking of a different module.

4.1.2 Adaptation

When a module in the queue needs to be adapted (fig. 28, right) to changes such as different clothes, physical or mental state of the user, this is also done via bit-vector masking, since the masking does not destroy the original classification capabilities of the classifier [31]. Also, the bit-vector masking is easily removable if the original classifier needs to be restored. To identify the bit-vector for the adaption of the respective module, data to adapt onto needs to be collected, which clearly indicates the changes to the module. The data also needs to include data pairs for all of the classes, which are recognized by this module. As data for representation of the complementary class in the identification of the bit-vector, the complementary class training data of this module can be reused.

Again, if one module is modified with a bit-vector masking, there might be inference with other modules, too. An example of such a circumstance is displayed in figure 30. The procedure of the bit-vector masking is

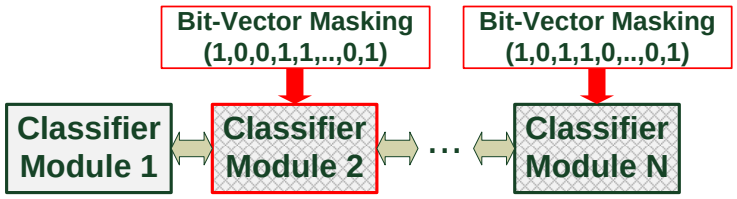


Figure 30: Example of adaptation of a classifier module onto changed conditions via bit-vector masking with additional masking of a different module.

again the same as described before and defined in subsection 3.1.6. If some of the other modules need to be modified according to the adapted module, which can be ascertained from the numbers in the confusion matrices, the approach is the same as for a module exchange. To the training data of the complementary class of the respective module, the adaptation data of the modified module gets added. This alters the capabilities of this module to distinguish between native classes and the complementary one.

As mentioned before, there are two fitness values for identifying good individuals due to the genetic algorithm. One is the mean squared error (MSE), which indicates the distance of the RFIS mapping outcome to the designated class mean. The other one is the percentage of incorrect classifications made for the masking with the individuals bit-vector. Both have advantages and disadvantages. The MSE results in an expressive reliability measure, but does not improve the classification accuracy directly. An improvement of the percentage of correct classifications can indirectly be improved through a filtering upon this more expressive reliability measure. The optimization on the percentage of incorrect classifications directly results in a more accurate classifier, but the resulting reliability measure might lose expressiveness. A later filtering therefore might not improve the classification results further. Both methods are evaluated in the evaluation section.

4.2 Challenge 2 - Extensibility

The addition of new classifiers to a preexisting activity recognition system is of high relevance to the topic and the second challenge presented in this thesis. The aim of the modular activity recognition is the common user, who has very individual demands and desires. Some of the users want to have certain sports or leisure time activities to be recognized, others need to have their work being part of the recognition. Considering all of these recognition needs in one system would lead to a high amount of activity classes to be supported. But, since the more classes that need to be detected the higher the complexity of the recognition system is and also the lower the accuracy, the aim is to reduce the amount of supported activities to a minimum.

If the whole activity recognition needs to be re-identified or retrained if just a few new activity classes are added, than the processing time for a server running the training process would rise significantly. Also, the user demands could only be temporary, since e.g. some sports can only be done during a certain season. In this case, the effort of training new recognition systems would rise again. Here the modular classifier architecture again presents certain opportunities to cope with the challenge of an **extensible** activity recognition system.

First, the problem of the addition of new classifier modules to a preexisting queue is analyzed. The dynamic queue is easily **extensible**, but the old modules might not recognize the transitions onto the new modules right away.

Last, a method is introduced, which overcomes the problem of faulty *complementary class* recognition without influencing the classification capabilities of the preexisting modules. Again a bit-vector masking provides the adaptation of the preexisting classifier modules, so they recognize transitions onto the new ones.

4.2.1 Problem Statement

As described in subsection 3.2.2 the classifier modules are arranged in a dynamic queue. Only the first module in the queue is active until it recognizes the complementary class. In this case, the modules successive to the first one get activated one after another until one classifies on a class besides the complementary one. This module gets then sorted in front of the queue and remains active until it recognizes the complementary class again. Since the capability of classifying on the complementary class is trained due to a novel training algorithm, which was presented in subsection 3.3, the recognition of transitions on newly added modules can not be included in the training phase, since these modules are not yet included in the training data. Therefore the problem arises, when new modules are added to a preexisting queue, the complementary class recognition of the old modules has no knowledge of the new modules and transitions on these new modules will never or rarely happen.

An example displayed in figure 31 and described in the following should further illustrate the problem:

1. A activity recognition system with N classifier modules is running on the user's mobile phone. At time $t = 0$ the module number 1 is first in queue, therefore actively classifying the incoming feature vectors.

2. At time $t = x$ two new modules are added to the end of the dynamic queue. These modules are trained in their complementary class onto each other and the preexisting modules, but the old modules in the queue have no link to the new modules.

3. A feature vector comes in at time $t = x$, which should be classified by one of the new modules at the end of the queue. The first and active module of the queue successfully classifies the feature vector on the complementary class. The feature vector is handed over from module to module, but the last of the old modules, module N, wrongly classifies the feature vector on a class besides the complementary one. The module is put first in queue and remains active. The transition to the correct new module could not be made.

4. The module N misleadingly is the first and active module in the dynamic queue of classifier modules. It wrongly classifies the new incoming feature vectors or a repetitive reorganization of classifier modules in the queue occurs. If the case of oscillating modules in the queue eventuates, the activation of the correct new module is possible.

It is assumed, that for the training of new modules the training data of the old modules is still present in the database, so the data can be added to train the complementary class of the new modules.

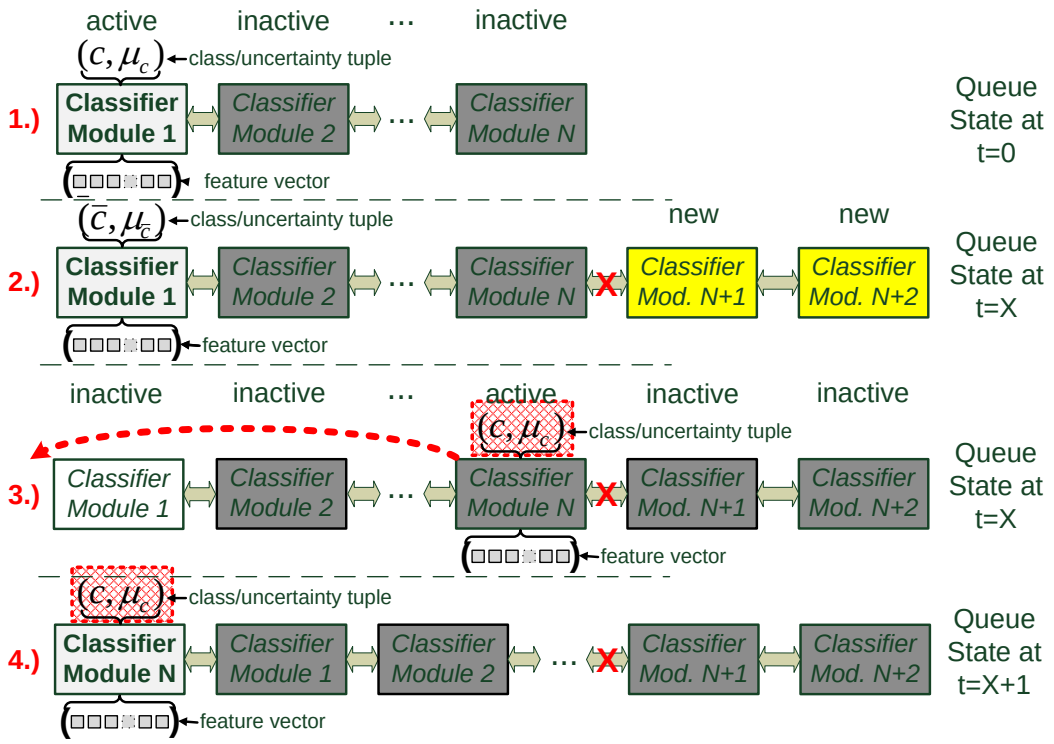


Figure 31: Addition of new modules to existing classifier queue.

4.2.2 Solution

The modular activity recognition system presented in this thesis is generally **extensible** without a retraining of classifier modules. In particular, the preexisting classifier modules need to be masked with a bit-vector to recognize transitions onto the newly added modules. The theory about how the masking works was presented in subsection 3.1.6. The practical application of the bit-vector masking for the **extensibility** is described in this subsection.

The solution to the problem of non-linked new modules in a dynamic queue is the bit-vector masking of the preexisting modules. When a new classifier module is added to an existing classification system the state transition in the dynamic queue needs to be modified. This transition is done when the active module $\mathbf{M}_i$ recognizes the complementary class $\bar{c}_i$. Since adding new modules presumes that no data for identifying the complementary classes of the old modules is available, the old modules are not able to detect a transition onto the new modules. This means that the new modules rarely get activated and therefore have little chances to be part of the classification.

If new classifier modules are added to the existing queue, the old ones need to be adapted so that they recognize transitions to the new classifiers. Through an adaption technique, in which the dimensions of each rule for each pre-existing classifier module are *activated* or *deactivated*, transition capabilities improve without changing the classification accuracy significantly. The original classifier module is preserved and can be restored at any time. The optimal combinations of *active/inactive* rule dimensions is searched for with a genetic algorithm, since the full search has an exponential runtime. The data for optimum search is the training data that the new modules were identified on, combined with the original training data.

The adaptation is done via a bit vector, that specifies the *active* and *inactive* dimensions of each rule for one module $\mathbf{M}_i$. Therefore the bit vector $bit_{\mathbf{M}_i}$ for module $\mathbf{M}_i$, which has n input dimensions and m_i rules, has a length of $b_i := n \cdot m_i$. To use the bit vector, an interpretation function $I(\mathbf{M}_i(\overrightarrow{v}_t), bit_{\mathbf{M}_i})$ is defined, that *switches* the rules dimensions temporarily, without changing the module $\mathbf{M}_i$ permanently. The interpretation function I is defined as a function mapping a module $\mathbf{M}$ together with a bit vector $bit_{\mathbf{M}} \in \{0,1\}^*$ of appropriate length to a module $\mathbf{M}'$. The bit-vector masking approach was described in detail in subsection 3.1.6 and how the bit-vector is identified was

described in 3.3.7. The training data, on which the bit-vector for each of the old modules in the queue is identified, composes of the original training data and the data which was used for training of the new modules. The training data of the new modules is added to the training data of the complementary class and therefore labeled zero.

To determine the respective classifier module's bit vector, a genetic algorithm is used. The space that has to be searched is 2^{b_i} for module $\mathbf{M}_i$. A complete search would therefore have a runtime of $O(2^{b_i})$, which makes it impossible to calculate in a reasonable amount of time. According to experience, the genetic algorithm can find a suboptimal, but appropriate solution in a time span that is acceptable and mostly much faster than retraining the old modules. Nevertheless, the search space can be limited. Here, the overlapping of membership functions can indicate which dimensions should be *deactivated*, so overlapping is reduced or even eliminated. This circumstance is not analyzed in this thesis and therefor is part of future work.

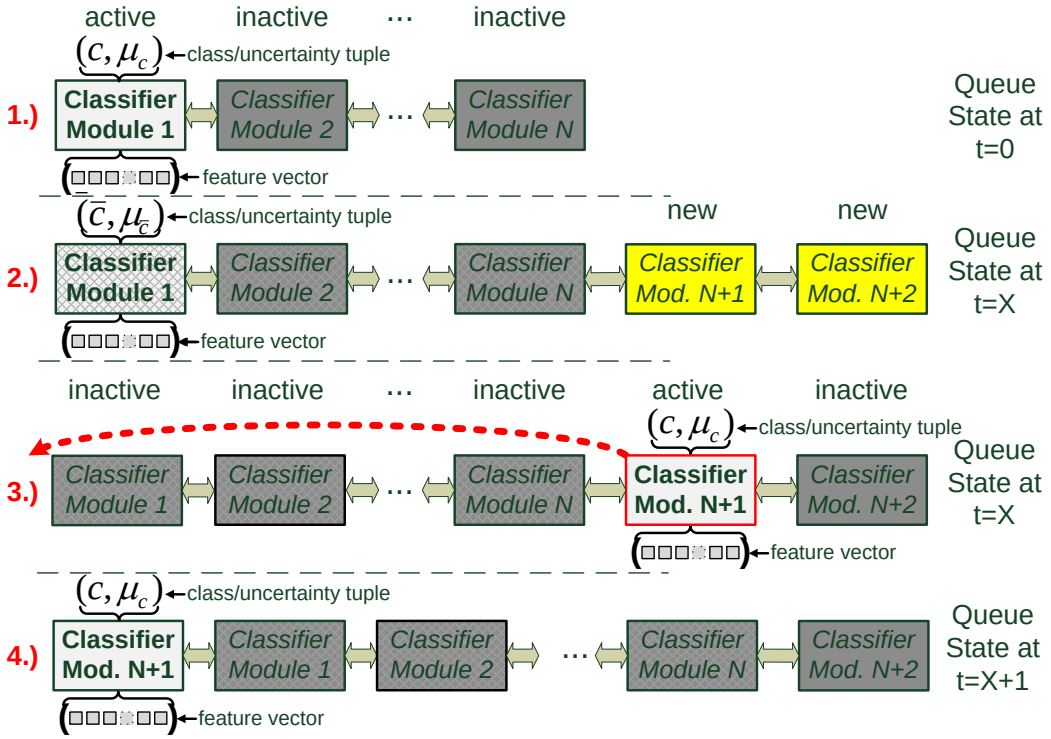


Figure 32: Addition of new modules to existing classifier queue with bit-vector masking of old modules.

The dynamic queue with bit-vector masked old modules behaves now according to the example displayed in figure 32 and described in the following:

1. For time $t = 0$ nothing changed, the dynamic queue still has N modules and module $\mathbf{M}_1$ is the first and active one.

2. At time $t = x$ two new modules get added to the dynamic queue. With the new modules bit-vectors are provided to the preexisting modules. These bit-vectors are used to mask the old modules so they recognize transitions onto the new ones.

3. A feature vector is incoming, which should be classified by one of the new modules. The old modules successively classify on the complementary class, because the masking takes effect. The module with the number N, which had recognized a class besides the complementary one in the example before, also classifies on the complementary class. Finally the correct module $\mathbf{M}_{N+1}$ gets activated and correctly classifies the feature vector. This module gets put first in queue and stays active until it classifies on the complementary class again.

4. The new module $\mathbf{M}_{N+1}$ is the first and only active module in the queue. This new module is now successfully linked in the queue. Same applies for the second new module in the queue.

In subsection 5.2 this approach of masking old modules in a queue that has been extended is evaluated. In these experiments a queue with four modules is **extended** by two more modules. The results of an **extension** are compared to a queue where all six modules are trained together.

4.3 Challenge 3 - Robustness

The **robustness** of a activity recognition is of high relevance and the third challenge presented in this thesis. Since the mobile phone is not firmly attached to any position on the users body, the sensor data on which the activity recognition is done is very noisy. Also the device can shift e.g. in the users pocket and therefore the sensor data changes. In this subsection two methods are discussed, which help increasing the **robustness** of the classification process.

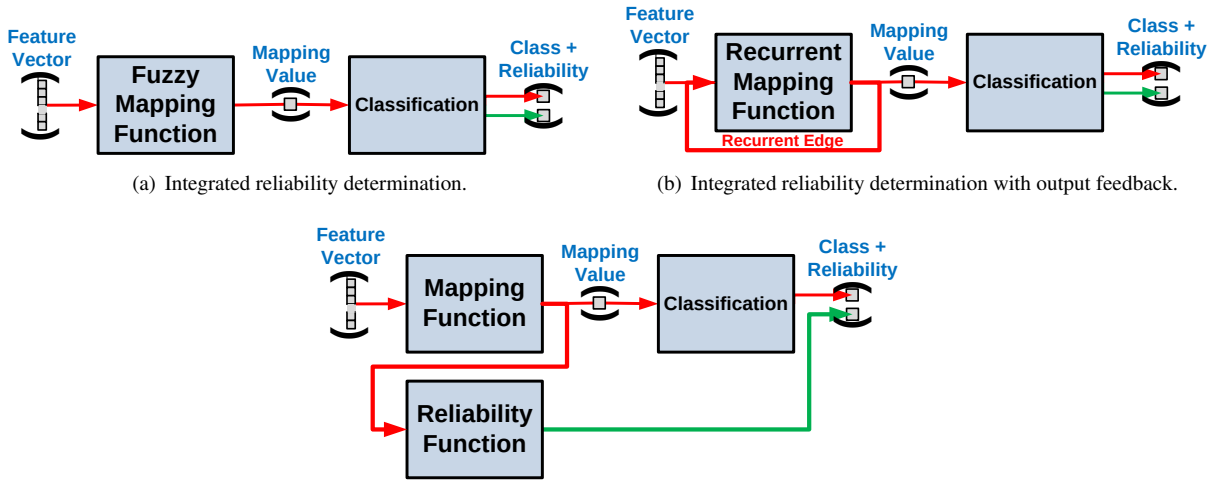
First, the various methods of deriving a reliability measure are listed. The methods are: (1) a parallel function for reliability determination, (2) an integrated reliability determination and (3) an integrated reliability determination with recurrence. The method of choice is the integrated reliability determination with recurrence, as defined and described in subsection 3.1.3, since it has the lowest overhead and the reliability measure is nearly as expressive as with parallel determination.

Last, the direct effects of a recursive mapping function on the **robustness** of the classification process is discussed and what the reasons for this circumstance are. This **robustness** relies on the frequency in which the users activities change. Also, the drawbacks of this approach are listed and discussed.

4.3.1 Methods Deriving Reliability Measures

With a mobile phone as sensor source the quality of the classification results vary strongly in reliability. Since the phone is not firmly attached to any body part, the data is very noisy and can not uniquely be classified. Some activities are more clearly identifiable, others are only in certain circumstances detectable. An activity recognition system that provides classification on every incoming feature vector does not consider this circumstances, but if a reliability measure is provided with each classification, the negative effects can be compensated. According to a reliability, the possible correct classifications can be separated from probably incorrect, thus improving the overall accuracy. Also, a next step in sensor data processing, such as a reasoning, could use this reliability to improve aggregated contexts from different sources as was demonstrated in [23].

There are three general methods of computing a reliability measure in an activity recognition system. These three methods correspond to possible system architecture styles or classes for reliability measurement, as shown in figure 33. Which of these styles are suitable depends on the classification method, but also on the specific setting in an application context.



(a) Integrated reliability determination.

(b) Integrated reliability determination with output feedback.

(c) Parallel function for reliability determination.

Figure 33: Different system architectures for determination of reliability measures of activity classifications.

Parallel Reliability Determination The most general architecture style is a parallel reliability determination (fig. 33(c)). Here a context classifier works in parallel to a reliability detector. The function that calculates the reliability measure has the same feature vector as input as the classification system. Furthermore the output of the mapping function used in the classifier is used as input. Therefore with this method the reliability function can be trained on the input/output behavior of the classifier. The reliability function thus behaves like an independent observer that constantly evaluates the output of the context classifier.

Such systems are especially useful when methods for classification and evaluation of the classification differ. No knowledge about the internal structure of the classification system is needed, only for training a data set of feature vectors and classifier mapping function outputs is needed, where each data pair is labeled according to the correctness of the classification. [27] shows that this approach is very beneficial for filtering contexts. This method delivers a reliability measure which includes several kinds of reliability information: (1) systematic errors - which kind of feature vectors and according classes have what kind of systematic error [30]; (2) input/output behavior - how behaves the classifier for which activity class; (3) quality through observer - independent observer's knowledge is included in the reliability function.

A disadvantage of this approach is the additional calculation effort for the computation of a second reliability function. On each feature vector not only the classifier, but also the reliability function has to be computed. Also, a parallel reliability function needs another training process, which doubles the processing on a server if the classifier is not preexisting. Since in this thesis the activity recognition has to be resource gentle for both, the mobile phone as well as the server - longer training periods lead to higher delays for the provision of classifiers to the users devices -, these two drawbacks are crucial. An integrated approach is preferred over a parallel one.

Integrated Reliability Determination A more compact method is a classification which provides a reliability measure itself (fig. 33(a)). An example implementation of this architecture style are Fuzzy Inference Systems (FIS) [29]. Here, fuzziness from within the mapping FIS can be used to derive the reliability level. E.g. in an TSK-FIS the outcome requires interpretation of the mapping outcome using a membership function. These membership functions provide an accuracy measure of the mapping, where a mapping close to the class center is of higher reliability. Is the mapping outcome close to a border of two classes, the decision for one of the classes is of lower reliability. Classes with high overlapping in clusters of feature vectors are not as easily separable as classes with no intersections. Therefore, the fuzziness of the mapping function provides the reliability of the classification.

With this method only the classifier has to be computed, where the reliability measure is a side product of the feature vector mapping. No additional parallel reliability function has to be calculated, nor an extra training process has to be computed on a server. Also, an observer has not to provide additional information about the classification process.

The disadvantage of this method is that only the fuzziness of the mapping model can be detected, with only small variations among different classes. The reliability of the system configuration itself will thus not be taken into account when calculating the reliability measure. Also, the extra knowledge provided through the observer and information about systematic errors can not be trained into the reliability function. The reliability in detection of the different activity classes is also not considered in the computation of the reliability measure with this method.

Integrated Reliability Determination with Output Feedback The third architecture style is the integrated reliability detection with output feedback 33(b). With this method, the output of the mapping function at $t = x$ is fed back as additional input at time $t = x + 1$. Therefor not only the internal reliability of the accuracy of the mapping function due to the use of fuzzy logic is reflected, but also the class dependent reliability. As mentioned before, some activities are more clearly detectable than others, which has a huge impact on the classification reliability. The feedback of the output as additional input has also a stabilizing effect, which is explained in detail in the next subsection.

The integrated reliability detection with output feedback is a combination of the solely integrated and the parallel reliability detection. A Recurrent Fuzzy Inference System (RFIS) is used in the novel modular activity recognition concept presented in this thesis. A RFIS as a mapping function in an activity classification process was presented in [23] and defined in detail in subsection 3.1.3.

The biggest advantage of this kind of reliability determination is, that the effort of deriving such a measure is only marginally higher compared to a non recurrent classifier. Also, since no additional function has to be

determined, the effort for training such a recurrent classifier is not much higher than training a non recurrent one. Although, the recurrence needs additional iterations of the training algorithm, which was described in subsection 3.3. Since the recurrence also improves the classification accuracy, a significantly longer training process as e.g. needed for a back propagation training [81] can be prevented.

The reliability measure derived through a RFIS mapping function includes two kinds of information, (1) the accuracy of the mapping process and (2) the class dependent reliability. Since the recurrence also stabilizes the classification process, the reliability is the lowest inbetween two different activity classifications. Here the classification process has to stabilize on a new activity, which results in more incorrect classifications and therefore in a lower reliability measure.

Nevertheless, despite these significant advantages the derivation of a reliability measure with a recurrent mapping function has also some disadvantages. Since no parallel system is used, which was trained according to the global knowledge of an observer, the reliability measure does not reflect that information. Also, the recurrence interferes with scheduling mechanisms, which is described in subsection 4.6.2.

4.3.2 Robust Recurrent Classification

As mentioned before, the recurrence in the RFIS mapping function does not only deliver the desired reliability measure, but also stabilizes the classification process. Therefore, the modular activity recognition is **robust**. Why the feedback of the mapping result as additional input makes the classification **robust** is analyzed here and evaluated in subsection 5.3.

Activity Change Frequency Using a classifier that has an output on every incoming feature vector, the process always has to deal with outliers inbetween correct classifications. Since the activity recognition presented in this thesis should also recognize activities where the steady orientation of the phone is important, like e.g. user is *sitting* or *standing*, a time series classifier, as e.g. was used in [151] and [140], is not applicable. Although, classifiers that analyze whole time series of feature vectors and not single ones would be invariant to single outliers.

Another method to cope with this problem is feasible, a feedback of the mapping outcome, which is used in this thesis. Here the circumstance is of benefit, that the frequency of human activity changes is much lower than the frequency of classifications. With the OpenMoko Freerunner phone an average classification rate of 12.5 classifications per second is the upper limit, whereas for the other platforms supported so far, iPhone and Android phones, the average rate of classifications per second is with $2.5 - 4.125$ is much lower. A course of actions where the human activities change in less than seconds is considered on the other hand as very rare. Nevertheless, this can happen, but since it is not very likely, this case is not considered in the architecture of the activity recognition system with recurrence. Therefore, a RFIS is used as mapping function in the activity recognition, which has a memory of one previous mapping.

Robust Classification A feedback of the mapping outcome can improve the **robustness** of a classification system significantly. Here the mapping value of a certain time $t = x$ is added to the feature vector at $t = x+1$ as additional input dimension. When the recognition starts at $t = 1$ there is no mapping outcome to be fed back, since at $t = 0$ the system did not begin to run yet. Since the mapping function does expect an input, a value has to be provided which does not interfere with the classification process, nor favors one class over another one. Also, if the feedback parameter is any particular class, the first module in the dynamic queue would be pushed to classify the data onto the native class which this parameter identifies. In case of the wrong module leading the queue, this would result in a faulty state of the queue and in a wrong classification. The correct state of the queue would then only be reached after a certain amount of classifications. The complementary class with the identifier zero is chosen to be the first feedback, since it does not identify any particular class or classifier module. Experience shows, that the queue stabilizes on the correct class and classifier module after the first classification.

For a classification at time $t = x$ of the first module in the queue on the complementary class, the feedback value is not changed, since the same feature vector is processed by the following module in the queue. If all modules classify on the complementary class, the feedback is the RFIS mapping outcome of the last module in the queue. Therefore, the feedback is as close to neutral as it was at the start of the classification. Also, the reliability value has some special considerations for a module alteration in the dynamic queue. The reliability measure for every

classification is multiplied until the correct module is found. This reliability is then the output of this classification and therefore reflects the reliability of all classifications until the possibly correct choice was made.

With this recurrent mapping function, the most incorrect classifications are made inbetween the activities. Does the user change from one to another activity, the classifier has to stabilize onto the new class. With a non recurrent classifier this happens immediately, since it has no memory. The classifier with output feedback needs to shift slowly from one class to another, the more memory of the past the slower the switch. Since in this case the memory of only one past mapping is present to the classifier, the switch is done in a few classifications, but with a high frequent activity change, the positive effects of **robustness** are nearly eliminated. The assumption is, that a normal user in an every day setting does not change their activities too quickly. If the user e.g. is *sitting* in front of a screen, she does that for a while and does not jump up to *standing* every other second. The longer the periods without activity change are, the better the effects of the RFIS mapping are on the **robustness** of the modular activity recognition.

4.4 Challenge 4 - Resources

One of the biggest challenges performing activity recognition on mobile phones are the limited **resources**. The mobile CPUs are not the biggest problem, since they increased in speed according to Moore's law, it is the limited capacity of the battery. Since the user is demanding a phone which allows the usage without limitations, the activity recognition should not constrict this usage. Also, the battery runtime should not be noticeably lowered, so normal usage cycles of the mobile phone without activity recognition should also be reached with.

There are two ways to cope with this challenge, one is to reduce the calculation effort for the accelerometer based activity recognition and the other one is to schedule the recognition, so it is not constantly using the processor. With the modular activity recognition the average calculation effort towards a monolithic classifier is not only reduced, but also the accuracy of the classification process is increased. The reduction of processing time for the activity recognition not only increases the possible processor load for the usual mobile phone applications, but also reduces the amount of energy expenditure due to the classification process. Also, due to the modularity scheduleable units are available, which can be activated or deactivated on demand. These two methods are interwoven since the scheduling of the modules is also used to reduce the calculation effort. The scheduling of the whole recognition process can further be used to switch off the classification process, so the usual power savings mechanisms can be reused.

First, the mechanism to reduce calculation time for the activity recognition is explained. Therefore a statistical model is established, which helps to find the best tradeoff between the modular classifiers accuracy and the grouping of the activity classes per module. Also, the overhead due to the calculation of the *complementary class*, which is used to detect the transition from one module to another, is discussed.

Second, the energy consumption of the modular classifier is analyzed. Due to the limited effort for the modular activity recognition compared to a monolithic classifier the energy consumption could be reduced to a low level. Due to the inability to activate power savings mechanisms the energy consumption could still be high. For this reason a very simple sleep time scheduling for the activity recognition is introduced, which schedules sleeping phases for the classifier queue as a whole. Since the focus of the thesis is the activity recognition and not scheduling mechanisms, the various scheduling techniques possible with a modular classifier concept are not listed.

4.4.1 Processor Load

A challenge in doing activity recognition on mobile phones is the processor load. Since the activity recognition is intended to run in the background serving information to applications and the user is focusing on the usual phone applications, the limitations of these due to the activity recognition would affect the acceptance of the recognition. Also, indirectly through the processor load the battery runtime is influenced, which also has an impact on the user acceptance.

Nowadays developments indicate that the performance of mobile CPUs will rise according to Moore's law, the power consumption of the processors will be lowered over time and the battery capacities will rise. This positively affects the available **resources** for the activity recognition, but the architecture and design of the recognition software has to do its share, too.

Due to this reasons, an architecture is needed that reduces the calculation time for each classification, but does not lower the statistical accuracy of the recognition process. With the modular classifier approach presented in this thesis, not only the calculation effort can be reduced compared to a monolithic architecture, but also the accuracy can be improved significantly. As the experimental evaluation of this statement follows, first the assumption has to be theoretically analyzed.

Modeling and Optimization of Classification Probabilities Since the arrangement of activity classes per module and the amount of modules is mostly arbitrary, a cost and effort function could indicate good solutions. Probabilistic modeling can help maximize the statistical classification correctness. The underlying FIS mapping function - due to the simplicity of the resulting probabilistic model the non recurrent mapping function is used - which provides a reliability measure for each classification that is valuable when interpreting a single classification in an application context or further reasoning. When quantitatively optimizing the system the fuzzy domain should be left in favor of probabilistic modeling, as not a single classification is looked at any more, but multiple samples. In this section therefore a first probabilistic formalization is provided based on experimental findings - presented in

the evaluation section especially in 5.4.1 - to explain the relationship between recognition rates, class grouping and different classifier sequences. The statistical model and optimization strategy which is defined in the following was presented before in [29].

In order to model the overall statistical performance of the proposed queued interpretation system, the probability for a correct classification $\mathbb{P}(\hat{x}_t = x_t)$ is modeled, i.e. that the real state x equals the classified state $\hat{x}$. Both random variables are defined over the same set of $\mathcal{C}$. In the dynamic scheme the classification queues are optimized on the previous state, which need to be analyzed separately:

$$\mathbb{P}(\hat{x}_t = x_t) = \sum_{\hat{x}_{t-1}} \left(\mathbb{P}(\hat{x}_{t-1}) \sum_{x_t} \mathbb{P}(x_t|\hat{x}_{t-1})(\mathbb{P}(\hat{x_t} = x_t|x_t, \hat{x}_{t-1})) \right)$$

To reflect the different precomputed classifier sequences a total order relation $<_i$ is introduced. The classifier queue $<_i$ orders $(\mathbf{M}_a <_i \cdots <_i \mathbf{M}_z)$ by their execution precedence. As in this thesis's experiments each classifier module $\mathbf{M}_i$ in the queue classifies onto a specific set of classes $\mathcal{C}_{\mathbf{M}_i} \subset \mathcal{C}$, so that $\mathcal{C}_{\mathbf{M}_i} \cap \mathcal{C}_{\mathbf{M}_j} = \emptyset$, or onto the *complementary class* $\overline{c}_i$. For convenience a helper function is additionally defined, that selects the index of the classifier responsible for a given class: $m(c) = (i|c \in \mathcal{C}_{\mathbf{M}_i})$. Further the random variable $\mathbf{M}_{x,t}$ is used to denote the classifier module's result at time t.

In the proposed activity recognition system the classifications inside a queue are only based on the current state x_t, while the choice of the queue depends solely on the previously classified state $\hat{x}_{t-1}$ via a static lookup. If the state $\hat{x}_t$ is classified by a matching on non $\overline{c}_i$ classifier result $\mathbf{M}_{m(x_t)}$ in the next steps the classifiers are in a sequence according to $<_{\hat{x}_t}$ executed.

As discussed before a classifier in a split classifier system can only classify correctly if the complementary state is classified for all previous classifiers. If this process takes into account the true positive probability, then the recognition rates of the classifiers and the queue system state can be calculated as follows:

$$\mathbb{P}(\hat{x_t} = x_t|x_t, \hat{x}_{t-1}) = \mathbb{P}(\mathbf{M}_{m(x_t)} = x_t|x_t, \hat{x}_{t-1}) \prod_{i|\mathbf{M}_i <_{\hat{x}_{t-1}} \mathbf{M}_{m(x_t)}} \mathbb{P}(\mathbf{M}_{i,t} = \overline{c}_i|x_t, \hat{x}_{t-1})$$

Because the internal classifiers are stateless - the stateless FIS mapping function instead of the RFIS is used in this model -, i.e. independent of $\hat{x}_{t-1}$, a model for the overall recognition performance based on the recognition rates, the class grouping and the classifier sequences is finally obtained:

$$\begin{aligned} \mathbb{P}(\hat{x}_t = x_t) \quad = \quad & \sum_{\hat{x}_{t-1}} (\mathbb{P}(\hat{x}_{t-1}) \sum_{x_t} (\mathbb{P}(x_t|\hat{x}_{t-1}) \mathbb{P}(\mathbf{M}_{m(x_t)} = x_t|x_t) \\ & \prod_{i|\mathbf{M}_i <_{\hat{x}_{t-1}} \mathbf{M}_{m(x_t)}} \mathbb{P}(\mathbf{M}_{i,t} = \overline{c}_i|x_t))) \end{aligned}$$

This model can now be used to derive optimization strategies for designing classifiers. However, because in theory every classifier function can be represented by an infinite number of rules in the TSK-FIS [146], the execution complexity of the TSK-FIS has to be considered when optimizing classification. To model this trade-off, a composite cost function that incorporates the amortized recognition rates and **resource** usage is introduced. The proposed optimization function

$$\text{minimize } \textit{cost} = E[\textit{class\_cost}] + E[\textit{exec\_cost}]$$

is composed by the expectation of false classification costs and the expected number of rule execution costs. $\mathbb{P}(\hat{x_t} = x_t|x_t, \hat{x}_{t-1})$ is used, which is associated with $\textit{class\_cost}_1 = 0$ costs and the inverse $\mathbb{P}(\hat{x_t} \neq x_t|x_t, \hat{x}_{t-1}))$, which associate with an application specific cost $\textit{class\_cost}_0$ to define the expectation as follows:

$$E[\textit{class\_cost}] = (1 - \mathbb{P}(\hat{x_t} = x_t|x_t, \hat{x}_{t-1}))\textit{class\_cost}_0$$

Because of the strictly compositional setup of the constructed FIS the execution time is proportional to the number of rules. In order to obtain the expected execution costs, the equation of subsection 5.4.1 is generalized in

a similar way as the equation for the recognition rates based on the number of rules in a FIS $|\mathbf{M}_i|$ and a cost factor *rule_cost*:

$$E[\textit{exec\_cost}] = \textit{rule\_cost} \sum_{\hat{x_t-1}} \sum_{x_t} \mathbb{P}(x_t|\hat{x}_{t-1}) \sum_{\mathbf{M}_i} |\mathbf{M}_i| \prod_{(\mathbf{M}_j r(\hat{x}_t) \mathbf{M}_i)} \mathbb{P}(\mathbf{M}_{j,t} = \bar{c}_i|x_t)$$

With a given algorithm for implementing classifier modules $\mathbf{M}$ like with the fuzzy system as an internal classification algorithm, the class mapping m and queue mapping r can be optimized to achieve a better recognition rate. Both the number of classifiers and the number of rules in a classifier can additionally be adapted. Resulting from this modeling one can easily see the interconnection of classifier grouping and recognition rates via the state and the transition probabilities.

Because the true positive rates of the classifiers are dependent on an initial grouping and the transition probabilities $\mathbb{P}(x_t|\hat{x}_{t-1})$ are again recursively dependent on the recognition rates, a highly non-linear optimization problem is resulting. This problem can be solved heuristically based on expert knowledge, or an automated design space exploration using empirical data can be done. This method is not used in the remainder of this thesis, since it demands an extensive design space search and does not contribute to the evaluation of the proposed modular activity recognition.

Complexity of Activity Recognition The complexity of calculating the activity recognition is divided in three parts, the feature extraction, the mapping function calculation and the fuzzy classification. Both, the feature extraction and the fuzzy classification need only a fraction of the computing time compared to the mapping function. The feature extraction (see subsection 3.1.2) was described before as very efficient, since the mean calculation (eqn. 2) only needs w additions for a window size of w sensor measurements and one division. For calculation of the variance (eqn. 3) the results of calculating the mean feature can be reused. Then the variance consists of w subtractions, w multiplications and one division.

Calculating the fuzzy classification (see subsection 3.1.4) needs the least effort. There only o triangular fuzzy numbers (eqn. 16) need to be calculated and the maximum over this o memberships needs to be determined. A triangular fuzzy number is a non steady function with a distinction of four cases, where two are very unlikely and need no further calculation at all. The other two apply more often and need a maximum determination, two subtractions and one multiplication.

The majority of the calculation time is needed to compute the RFIS mapping function, since it uses various matrix multiplications and exponential function calculations. As the model established before describes, the calculation costs are strongly depending on the amount of modules, average number of module switches, the amount of rules per class and *complementary class*. This makes the calculation effort for the mapping functions in the dynamic queue hardly predictable, but the effort for calculating the output of one modules mapping function is predictable. For the computation of the RFIS's mapping function the biggest effort is the determination of the exponential function. The time to calculate this depending on the programming language and the processor architecture. Clear numbers for the exponential function on the bases of Euler's number are hard to find in the literature and can therefore mostly only be determined through empirical measurements. Each rule (eqn. 11) of the RFIS consists of a covariant membership function (eqn. 10) in the antecedent part and a linear function (eqn. 5) in the consequence part. Since the mapping function is a recurrent one, the input dimensionality it is operating on is not n but $n + 1$. Therefore, the calculation of the consequence part needs $n + 2$ additions and $n + 1$ multiplications. The more calculative intensive part of computing the RFISs is the antecedent part which includes one $n + 1$ vector matrix multiplication, one vector vector multiplication, one scalar multiplication and the most costly an exponential function calculation. Finally the weighted sum average (WSA, eqn. 12) is calculating the mapping functions output by multiplying the consequence with the antecedence of all rules, summing all products and dividing the result through the sum of all antecedents. This results in m multiplications, $2m$ additions and one division for m rules.

An overview of all the mathematical operations of which part of the modular activity recognition is given in table 2. Here all operations are categorized in $+$-, $\cdot$-, $\div$-, $\max$-, switch- and exponential operations. Each part of the processing queue is listed till after the classification. Also, a sub categorization is made according to the function analyzed. In the end a best and worst case estimation is made for the whole modular activity recognition. The number of rules per mapping function is therefore denoted with m_i, for the number of classes per module o_i and the number of modules n_M.

Part	$+$	$\times$	$\div$	$\max$	switch	e^x
Feature Extraction $\overrightarrow{\mathbf{v}}_t$						
Mean	w	-	1	-	-	-
Variance	w	w	1	-	-	-
Summary	$6w$	$3w$	2	-	-	-
RFIS Mapping $\mathbf{S}(\overrightarrow{\mathbf{v}}_t)$						
Antecedent	$m(n^2+2n)$	$m(n^2+3n+3)$	-	-	-	m
Consequence	$m\cdot(n+1)$	$m\cdot(n+1)$	-	-	-	-
WSA	$2(m-1)$	m	1	-	-	-
Summary	$mn^2+3mn+3n-2$	$m((n+2)^2+1)$	1	-	-	m
Fuzzy Classification $\mathbf{K}\left(\mathbf{S}(\overrightarrow{\mathbf{v}}_t)\right)$						
Membership	$2o$	o	-	-	$4o$	-
Classification	-	-	-	$o+1$	$o+1$	-
Summary	$2o$	o	-	$o+1$	$5o+1$	-
Overall Activity Recognition (Dynamic Queue)						
Worst case	$6w+3n_Mn-2n_M+\sum_{i=1}^{n_M}2o_i$ $+\sum_{i=1}^{n_M}\left(m_i(n^2+3n)\right)$	$3w+\sum_{i=1}^{n_M}o_i$ $+\sum_{i=1}^{n_M}\left(m_i(n+2)^2+1\right)$	$2n_M+4$	$\sum_{i=1}n_M(o_i+1)$	$\sum_{i=1}n_M(5o_i+1)$	$\sum_{i=1}n_Mm_i$
Best case	$6w+2o_i+3n-1$ $+m_i(n^2+3n)$	$3w+o_i$ $+m_i(n+2)^2$	6	o_i+1	$5o_i+1$	m_i

Table 2: Overview mathematical operations of the different parts of the processing queue.

The real effort for calculating the modular activity recognition lies somewhere in between the best and the worst case. This is depending on the activities which should be recognized and the grouping of the classes per module. If data has to be classified which demands many module switches, the effort is rising against the worst case. Also, if data is incoming which regularly can not be classified by any module besides on the *complementary class* the complexity will be close to the maximum.

Complementary Class an Module Switch Overhead One might think if the classes per module are successively reduced and the amount of modules is increased, the effort for calculating the modular activity recognition is decreased stepwise. In this manner the lowest effort of activity recognition should be a system where only one class is classified on per module. This is only partly true, since the overhead for classifying onto the *complementary class* has to be considered, too. Also, the overhead for the module switches in the dynamic queue has to be taken into account, which was not considered in the estimations before.

The amount of rules for the *complementary class* can empirically be determined for a varying amount of activity classes per module. Also, for a test set with a high amount of switches of successive activities, the average amount of evaluated rules can experimentally be determined. Evaluations will show, that the overhead for calculating the rules of the *complementary class* is not increasing with lowering the amount of classes per module. Although, the arbitrary or cost model based (derived before) grouping of activity classes per module will result in the loss of a meta semantic called *conditional context*, but the reaction on the different *conditional contexts* could be a solution for the challenge **conditionality**.

The overhead for the module switch is more difficult to measure, since this is depending on the implementation of the modular activity recognition, the programming language used, the compiler translation result and the processor architecture. Numbers can only be derived when executing the activity recognition on a firm set of test data, a designated total amount of activity classes and a varying number of rules per module. Since the overhead for switching the modules in the dynamic queue is considered to be negligible, but the effort of establishing such an experiment would be very high, this is not evaluated in this thesis.

4.4.2 Energy Consumption

The next part of the challenge **resources** is the energy consumption of the activity recognition. Due to the modular structure and the resulting reduced effort for the activity recognition compared to a monolithic classifier, the processor load was reduced to a minimum and therefore also the energy consumption. But there is more to reducing the energy consumption of an algorithm which uses sensors than minimizing the processor load. Since the sensors and the sampling of them also needs energy, reducing the calculation effort of the activity recognition does not effect

this power drain. Also, with an *always on* activity recognition the usual power savings mechanisms, such as turning off the processor and sensor periphery, can not be utilized any more.

First, the effects of an *always on* activity recognition is discussed in this subsection. What are the effects of an always running processor and sensor periphery. Second, a solution to this challenge is presented, a sleep time scheduling for the activity recognition. Due to the focus of this thesis, the sleep time scheduling is a very simple one only to indicate the possibilities in this direction.

***Always On* Activity Recognition** For a normal mobile phone usage there are always periods where nearly no power is drained. If the user is not actively using the mobile phone for making a call, surfing the Internet or any other application available on modern phones, then the processor and all the periphery can be switched off. Only the mobile radio unit is needed to keep the connection to the access point unless the phone is switched to flight mode. Since this topic is not in the focus of this thesis, other sources for getting some general numbers about power consumption are taken into account.

A complete analysis of the power consumption of the different parts of the OpenMoko Freerunner can be found in [38]. Different baseline cases are defined and analyzed. The suspended device needs the least energy. In this case the phone only needs to keep the communication processor running in a low level activity mode. The authors measured an average power consumption of 68.6mW when the phone is in *suspend mode*. Another mode analyzed by the authors is the *idle mode*, where the device is fully awake, but no application is active. The backlight was turned off, but the rest of the display subsystem was enabled. The average power consumption measured by the authors was with 268.8mW much higher compared to the *suspend mode*. As one of the biggest power drains the displays backlight was identified, which needs energy in the range of 7.8mW (minimum brightness) to 414mW (maximum brightness).

This thesis focuses on an activity recognition, which is always running in the background providing applications with classification information. The activity recognition generally claims a certain amount of processor load, therefore the CPU and the periphery can not be switched off for longer periods. The conclusion is that the energy consumption of the modular activity recognition will be somewhat higher than the *idle mode* described in [38]. Although, the *idle mode* in [38] includes the running display subsystem, which draws 80mW. The authors also measured the energy consumption of varying micro-benchmarks. As evaluation results in [63] show, the recognition of many activities needs less than 1.5% of the total available processing time. Since the micro-benchmarks used in [38] vary from high memory to high CPU utilization, the proposed modular activity recognition will need significantly less energy than all of the micro-benchmarks. The energy consumption of the micro-benchmarks varay from 160mW to over 220mW and the idle CPU drains about 60mW. Therefore the modular activity recognition, which does not need a high amount of memory, would at most be 80mW higher than the *idle mode*. This compensates the effect, that the display subsystem is not needed for the activity recognition running in the background.

Still, the energy consumption of the accelerometer sensor with a sampling rate of about 100Hz are not considered so far and are not analyzed in [38]. In [48] the energy consumption of sensor enabled applications on mobile phones are analyzed. Since the authors use a separate sensing device, which is connected over Bluetooth to the phone, the power drain due to the sensor device can not be isolated. Furthermore, there could not be any numbers derived for how much energy the OpenMoko Freerunner accelerometer periphery drains. It is assumed through experience with other sensing devices [62], that the energy drain due to the accelerometer sensory would not be a large portion of the activity recognition itself. Therefore, the energy consumption of the modular activity recognition on the OpenMoko Freerunner device is considered to be somewhere between 268mW and 300mW.

For the activity recognition running on a state of the art mobile phone with a floating point unit (FPU), the processor load would only be a fraction of the processing time needed on a Freerunner. Therefore the energy consumption would even be less. Nevertheless, the aim is the energy consumption in a *suspend mode*. To get closer to the energy consumption of the mobile phone in *suspend mode*, the activity recognition should not to be running all the time, but sleep phases need to be entered. In a sleep phase the normal *suspend mode* can be activated and therefore the power drain would be set to a minimum.

Sleep Time Scheduling Algorithm for Activity Recognition Since the energy consumption of the activity recognition can not only be limited due to reduced calculation effort, another method is introduced to limit the power drain. As analyzed before, the normal power consumption in an *idle mode* of a mobile phone is not increased

much due to the activity recognition. Since the phone can not switch to a *suspend mode*, which would drain significantly less energy compared to the *idle mode*, due to the *always on* activity recognition, a mechanism is needed to overcome this issue.

Depending on the frequency in which the user changes the activity, the recognition classifies on the same classes for short or very long periods. If the period is long, the activity recognition does not provide new information unless false classifications occur. In these periods, the activity recognition could be set to sleep, so the normal *suspend mode* can be activated. Therefore a huge amount of energy can be saved - [38] the difference for an OpenMoko Freerunner phone would be about 200mW -, which would increase the runtime of the battery and acceptance of an activity recognition by the user. The problem here is the length of the sleep phases which are scheduled. A very simple mechanism comes into mind, the frequency of activity class changes detected through the activity recognition itself.

With the modular activity recognition architecture, there are two entities possible for input of a sleep time scheduling algorithm. One is the frequency of different classes in successive classifications, the other one is the frequency of module alteration in the dynamic queue. Since the modular recognition architecture is different from the generally used monolithic methods, the frequency of class changes is used in a very simple sleep time scheduling algorithm. This method should therefore be applicable to nearly any activity classification method, since the whole classification process is considered a black box and only the sequence of classification outcomes are used for the sleep time scheduling.

After each activity classification it is checked, if the classification has the same result as the classification before. If the classification is the same, a counter is increased and a method is called that initiates a sleep phase. This sleep phase is an integer value of the time which is needed for sampling a window size of sensor measurements. Since the calculation time for one classification and feature vector calculation is less than for sampling one window size of sensor measurements, this method does not result in missing sensor values if no sleep phase is scheduled. If the old and the current detected activity class are not the same, the counter is set to zero and therefore no sleep phase is executed. A maximum integer value for the sleep phases prevents the counter to constantly be increased and therefore the sleep phases will be too long. The values for this maximum boundary needs to be determined through experiments, which is again out of focus of this thesis.

As mentioned before, the sleep time scheduling algorithm is very simple but effective, since it enables the recognition process to use the devices power savings mechanisms again. With an activity recognition that is always running without any pauses, the power savings mechanisms, such as a *suspend mode*, can never be active. Although, the sleep time scheduling is not in the focus of this thesis, the evaluation of this very simple sleep time scheduling is presented to close the gap in evaluation of the challenge **resources**.

4.5 Challenge 5 - Conditionality

The last challenge to be analyzed in this section is **conditionality**. Since a mobile phone can be everywhere in the environment or on the user, the individual support and reaction to the phone's **conditionality** is the only way to enable the activity recognition. This is due to the fact, that a different **conditions** imply different sensor patterns, which need different classifiers. Also, some **conditions** the phone is carried in allow the recognition of certain activities others imply other activities. E.g. if the phone is carried in the user's pants pocket, then the recognition of all leg based activities are possible, but if the phone is carried in an arm pocket, then the activities due to arm movement are able to be recognized.

Another possibility is to have the phone lying somewhere in the surrounding of the user, e.g. on a desk or in the car, which also implies different sensor patterns and therefore demands different classifier modules. A third possibility is to carry the phone in a known position e.g. in the pants pocket, but the user is currently in a special location such as in a bus, train or air plane. All these possibilities demand different classifiers due to very different sensor patterns, where the recognition of all these patterns with one classifier or few modules increases the complexity and lowers the accuracy of the activity recognition.

First, the **conditionality** is analyzed. The modular approach offers here the key advantage to offer a individual recognition for each of the different sensor patterns without a significant loss of accuracy or increased calculation effort. Last, the modularity is analyzed on the capabilities to react to different phone orientations where the **condition** remains static.

4.5.1 Conditional Context

The modularity of the activity recognition offers a unique approach to react onto **condition** dependent special sensor patterns. This is done without increasing the complexity of the recognition process significantly. The detection of the **conditionality** of the device or the user is called *conditional context*. This additional information about the conditionality enriches the recognized activities.

Module *Conditional Context* Relation As explained and analyzed in various parts of this thesis, the activity recognition is not done with one monolithic classifier, but with many classifier modules dynamically arranged in a queue. Only the first module in the queue is active at one point in time and remains active until it recognizes the so called *complementary class*. When this *complementary class* is recognized by the first module in the queue, the feature vector the *complementary class* was recognized upon is handed over to the next module in the queue. This process proceeds until one module is classifying the feature vector on an activity class besides the *complementary* one. This module is then moved to the first position in the queue and remains there until it also recognizes the *complementary class*.

This self scheduling of the queue is used to react on the *conditional context*, since only one module is actively classifying the incoming feature vectors and the decision of which module should be the first one is made by the modules themselves. To accomplish this the classes have to be assigned to each module according to the meta knowledge *conditional context*. In this manner e.g. all activity classes for the user having the phone in her pants pocket are grouped together in one module and for holding the phone in her hand in another module. So each module has a meta semantic and the currently active module implies the *conditional context*. A similar approach was traced in [133] also for activity recognition on mobile phones, but in [26, 25] this concept was published earlier.

An example is given in figure 34, where five *conditional contexts* are coped with by five classifier modules. The recognized *conditional contexts* in this example are mobile phone *in pants pocket*, *in shirt pocket*, *being held*, *lying on table* and *lying in car*. The currently detected *conditional context* is *in pants pocket*.

A further subdivision of the modules is possible, so more fine grained *conditional contexts* can be reacted on, so the complexity of the modules is not too high and the accuracy still reasonable. E.g. having the phone in the *pants pocket* could lead to a fairly high amount of classes to be detected with one module. The subdivision could here have more than one module recognize the *conditional context*, but also have further meta knowledge available of e.g. the user is *moving* or *calm*. Even more meta knowledge can be derived, if e.g. a module is detecting different dance styles, than the conditional contexts of this module being active would be *in pants pocket*, *moving* and *dancing*.

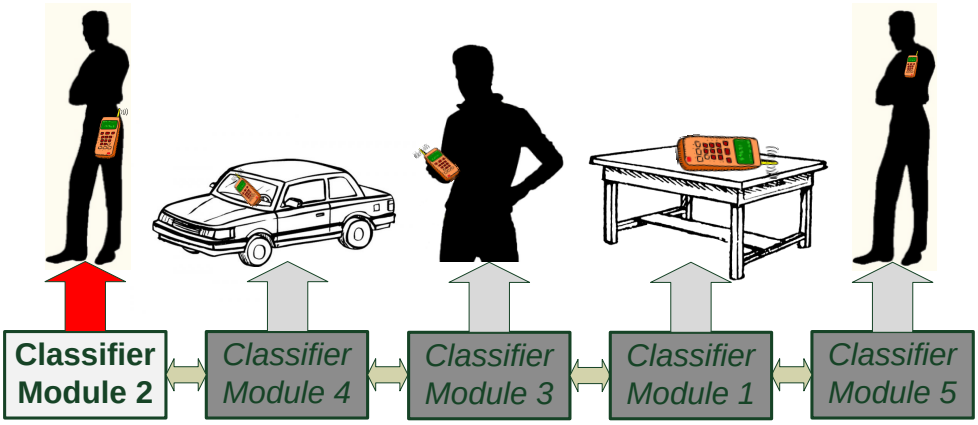


Figure 34: Example of different *conditional contexts* and respective modules.

Other Methods to Cope with Different *Conditionalities* Having a modular classifier structure with semantically grouped activity classes is not the only reaction possibility for the different *conditional contexts*. The mere classification results in the same knowledge, e.g. detecting activity *standing* results in having the phone in the *pants pocket*. This can be done with a monolithic classifier, but also with a modular recognition architecture, where the activity classes are not semantically grouped in the modules. Another possibility is to have an activity recognition, where every module is only recognizing one class besides the *complementary* one.

There are two main problems with the monolithic and the non semantically grouped modular approaches. On the one side, the accuracy for the detection of *conditional contexts* is mostly lower and on the other side, the complexity of the activity recognition will rise.

Having all classes classified by one monolithic classifier has certain disadvantages such as lower accuracy and higher complexity, which was analyzed in subsection 4.4. If one classifier has to classify all the different patterns from different locations and distinguish also different *conditional contexts*, the complexity of a monolithic classifier will rise and the accuracy will drop further.

If the activity classes not semantically grouped according to the *conditional context*, due to the overhead through module switches in the dynamic queue the calculation effort increases and the overall classification accuracy decreases. This is due to the fact, that modules which classify on semantically similar classes stay active for a longer period. Therefore, the modules do not have to be switched that often, which results in less average evaluated rules. An increased module switch frequency also effects the accuracy, since the more modules that have to be evaluated the higher the probability of incorrect classifications. E.g. if the activity classes are grouped according to their *conditional context*, the classes *sitting* and *standing* might be on the same module. When these classes are on different modules, then every time the user changes between this two activities the order of the modules in the dynamic queue has to be changed. The evaluation of this effect demands a data set that is originating from a normal progression of the user's activities and not a randomized data set as used so far. Since acquiring such a dataset is very difficult despite the various tools implemented for this purpose (see subsec. 3.4) a evaluation of this effect is not presented in this thesis, but is considered in the future.

The last possibility is to have an activity recognition, where each module is only classifying on one class besides the *complementary* one. Since the recognition rates of this kind of modular classifier queue are very high, as will be shown in subsection 5.4.2, the overhead due to the increased module switch must be compensated due to the better performance of each module. This approach surprisingly seems to be an alternative towards semantically grouping the classes per module, which are classifying one or more classes.

4.5.2 Phone Orientation

A specialty to mobile phones that is not covered through the term *conditional context* is that the device can have very different orientations. E.g. if the phone is in the *conditional context phone in user's pants pocket* the device

can generally have four different main orientations, if the phone fits the pocket crosswise the number is increasing to eight. Since the orientation of the phone is also used to distinguish between different static activities, such as *sitting* and *standing*, the mere orientation can not be excluded form the recognition process.

To cope with this problem, not only the respective *conditional context* is related to a module, but also the different orientations. Therefore, if the user has the phone e.g. *pants pocket* the phone needs four modules for this *conditional context* to react onto the four main orientations. This example is displayed in figure 35. For each of

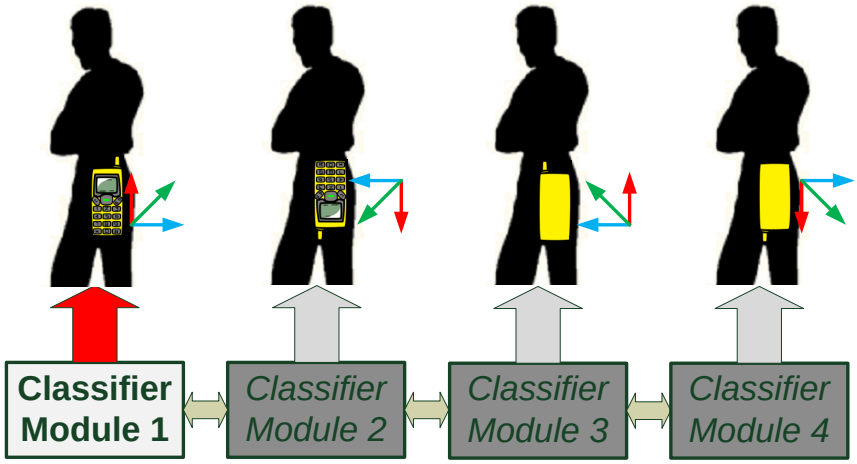


Figure 35: Example of different device orientation for the phone in user's *pants pocket* and module correlation.

the four different orientations the respective module selects itself in the dynamic queue. A similar approach was proposed in [133], but again earlier in [26, 25].

The main problem here is that different orientations also imply different activities. Such as if the phone is *upright* in the *pants pocket* and the user is *standing*, then the phone has the same orientation as when the user is *holding* the phone and is *making a call*. This contradiction can not be solved and both can not be recognized simultaneously. Here the user has to choose the reaction of either the orientation or the recognition of the respective activity. Since the evaluation of this approach of coping with the different orientations of a mobile phone increases the degree of freedom for the evaluation significantly, this is not part of the evaluation in this thesis.

4.6 Tradeoffs

kDuring these evaluations of the different challenges, some tradeoffs could be identified and some of the tradeoffs are not easy to analyze, such as the one between the challenges **flexibility** and **conditionality**. Here the design decision of grouping the activity classes per module according to the **conditionality** reduces the **flexibility** of the design. Since a good performance for the recognition of activities and *conditional context* is a modular recognition where each module only recognizes one class besides the *complementary* one, this tradeoff is obsolete. Also, as **flexible** an activity recognition was defined, where each module is individually adaptable or exchangeable and not as the **flexibility** of the design process.

Therefore, two tradeoffs were chosen to be exemplary for all the possible ones, since they offer a clear description and actual numbers for the tradeoff can be determined. One is the tradeoff between the activity classification accuracy and the *activity event* detection rate, where first the new concept of an *activity event* is explained. This tradeoff occurs due to reduced classifications due to the filtering.

The other one is the tradeoff between the challenges **resources** and **robustness**. The energy consumption of the activity recognition can be reduced due to the sleep-time scheduling, but this interferes with the recurrent mapping function of the classification process. As the sleep-time scheduling produces smaller or bigger leaps forward in time, the recurrent dimension can hold information about the wrong activity class.

4.6.1 Activity Classification Accuracy vs. Activity Event Detection Rate

Due to the filtering upon the reliability measure a tradoff between the overall statistical accuracy and the *activity event* detection rate occurs. The filtering does not only improve the overall recognition rate of the modular classification, but also reduces the amount of classifications passing the activity recognition. This filtering hits a critical point, when statistically less then one classification is passing per second. This is an arbitrary threshold and has no evidential background, that is why the term *activity event* is introduced. The *activity event* provides a value to empirically evaluate the impact of the filtering on the following application or reasoning process.

Activity Event An *activity event* is the period, where the user only performs one activity. It starts with the transition form one activity to the respective and ends with the change to another activity. In terms of successive classifications without gaps, the *activity event* starts when the first classification should be made on the respective activity and ends when the last one for this activity should be made, which is graphically described in figure 36. Since a classification process will never be one hundred percent accurate, there will always be incorrect classifi-

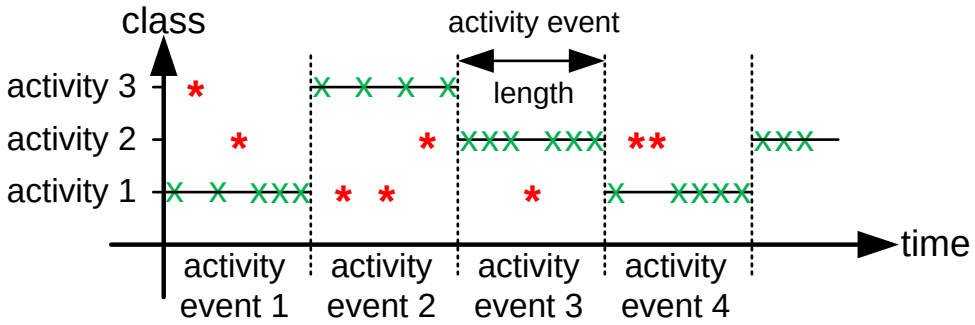


Figure 36: Example of *activity events* for three activity classes with correct (x) and incorrect (∗) classifications.

cations. A detected *activity event* is therefore an activity that is at least recognized correctly one time until a new activity is performed. If only incorrect classifications are made in one *activity event* period, this *activity event* is considered to be missed.

Due to the filtering upon the reliability measure the amount of passed classifications to an application or a reasoning is reduced, in some cases significantly. The implications for the overall statistical accuracy of such a filtering are very positive as will be shown throughout the whole evaluation section, but the *activity event* detection rate is dropping with an increasing filter threshold. This is due to the fact that not only true negative classifications

get filtered out, but also with every filtering upon a reliability measure a false negative rate occurs. A former correct detected *activity event* can be missed due to the filtering, if the correct classification of this activity is mistakenly taken as an incorrect one.

The tradeoff can be influenced through the filter threshold upon which is decided if the reliability measure is indicating a correct or incorrect classification. If the user's application or a reasoning process demands very high overall recognition rates, but a high *activity event* detection rate is not important, then the threshold can chosen to be very high. If the *activity event* detection rate is more important, then the threshold should be close to zero. The best tradeoff between accuracy and event detection is when both rates are equally high. This point is individual to every activity recognition and *activity event* period length, which can be read out the plots of accuracy and event detection which are shown in the evaluation section.

The problem of the *activity event* detection is even tightened if a sleep-time scheduling (subsec. 4.4.2) is added to the activity recognition. In this case the activity recognition is set to sleep for varying time periods. Therefore, the activity recognition could be set to sleep in one *activity event* and awake in the next one, thus reducing the amount of classifications per *activity event*. Also, the recurrence of the mapping function behaves badly due to such a sleep-time scheduling, which again results in a decreased accuracy and event detection rate of the activity recognition. The tradoff between the challenges **robustness** and **resources** will be analyzed in subsection 4.6.2.

The problems of filtering can be bypassed if the reasoning is fuzzy and the applications use only the output of the reasoning. There, the filtering is not done explicitly after the classification, but implicitly due to the reasoning with uncertainty. Therefore, the amount of passed classifications is not reduced and the recognition accuracy is improved according to the reliability of the reasoning process. An example of such an approach can be found in [23].

4.6.2 Robustness vs. Resources

The **robustness** of the modular activity recognition is reached due to the recurrent mapping function used in the classification process. The RFIS not only provides a stabilization of the classification, but also enables the derivation of a reliability measure. Upon the reliability measure filtering can improve the overall statistical accuracy of the activity recognition. Alternatively the reliability measure can be used in the application using the activity recognition information to make a more fine grained decision or in a reasoning with uncertainty to improve accuracy towards a crisp reasoning. Therefore, the recurrence of the mapping function is very relevant for the activity recognition, but combined with power savings mechanisms to limit **resource** consumption, the feedback of the output as additional input of the mapping function has certain disadvantages. If a sleep-time scheduling is introduced, which switches the whole activity recognition off for varying time periods, then the recurrent dimension of the mapping function input can contain false information. Thus the recognition process needs some classifications to stabilize on the new activity.

Recurrence vs. Sleep-Time Scheduling As described in subsection 3.1.3 mapping the incoming feature vectors onto a classifiable value uses a Recurrent Fuzzy Inference System (RFIS). There, the output of the mapping function is fed back as additional input, thus increasing the input dimensionality by one. This recurrent mapping has certain advantages as will be analyzed with the challenge **robustness** in subsection 4.3.

The sleep-time scheduling offers the opportunity to re-enable the suspend mode of the mobile phone, which is not possible with an *always on* activity recognition. This helps to reduce power consumption due to the modular activity recognition significantly, since the minimization of the power consumption due to the reduction of the processor load can only converge to the lower limit of the idle mode. The sleep-time scheduling was described in detail in subsection 4.4.2 and will be evaluated in subsection 5.4.3 of the challenge **resources**.

These two challenges collide now as previous experiments have indicated and the evaluation will show. Since the sleep-time scheduling causes leaps forward in time of varying length, the feedback of one mapping outcome can be originating in one activity, where the classification should be made for another activity. This circumstance is illustrated in figure 37 as an example. As can be seen in the upper half of the figure, the normal activity recognition has the most incorrect classifications when the user changes from one activity to another one. If the activity recognition should recognize a new activity, then the first few classifications are possibly incorrect since the mapping needs to stabilize on the new activity due to the recurrence. After these initial mis-classifications, the mapping is usually very **robust** and only a few mappings will be wrong.

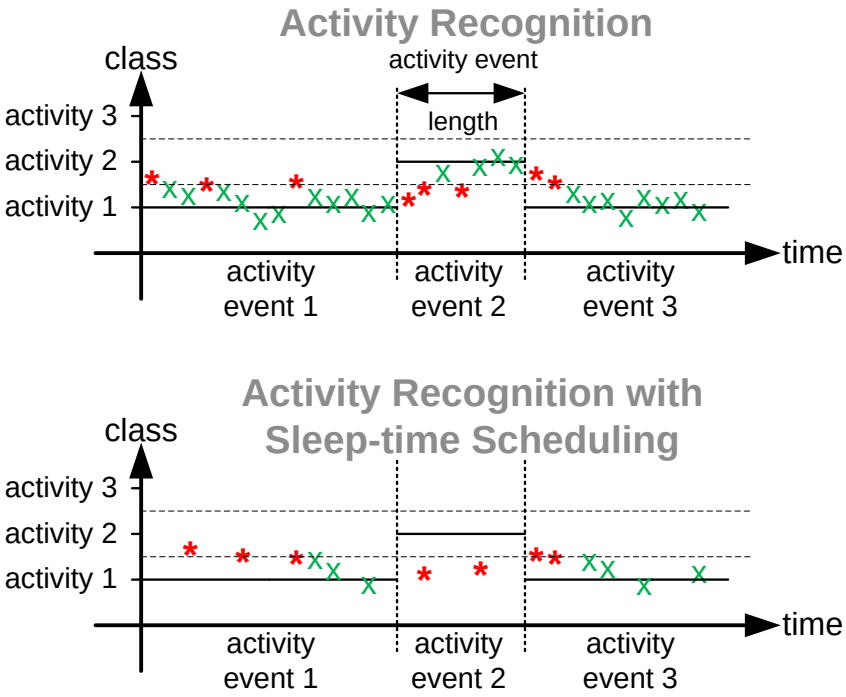


Figure 37: Example of activity recognition without and with activity recognition - RFIS mapping with correct (×) and incorrect (∗) classifications.

If a sleep-time scheduling added, then the average classification accuracy is lowered significantly. As illustrated in the lower half of figure 37, the classification process does not need to stabilize for much more successive mappings, but the sleep-time scheduling reduces the amount of classifications per *activity event*. In the example, the activity changes, but the classification still recognizes the previous activity, since the recurrent dimension of the mapping function input still carries the value of the old activity. As the length of the sleep phase is dependent on the frequency of the activity class change and this has not changed, the maximum length (in the example three classifications) is still active. So in the end, the mapping needs 13 classification phases to switch to the new activity, but effectively only three incorrect classifications were made. In the next *activity event*, which is much shorter in length, the mapping can not switch to the new activity. The switch occurs, when the event is already over, thus resulting in further incorrect classifications.

The tradeoff between **robustness** and **resources** can be influenced by adjusting the sleep-time scheduling. One method is to set the maximum sleep phase length to a minimum in order to reduce the amount of incorrect classifications. With a small maximum the energy saving would only be marginal, since the suspend mode can barely be activated until the processor needs to be reactivated for the next classification. Another possibility is to set the maximum sleep phase to a high value, so the suspend mode can be activated for longer periods. This is especially a problem, when the activity changes from a long event length to a short one. In the case of the long *activity event*, the sleep-time scheduling reaches the maximum sleep phase length. Then the new activity occurs, but due to the maximum length, the event is jumped over or only a few classifications are made in it, which is usually not enough for the mapping to stabilize on the new activity (fig. 37, lower graph).

Another factor that influences the tradeoff between **robustness** and **resources** is the length of the *activity event*, but this factor can not be influenced in real life. This factor has nearly the same influence on the tradeoff, since the length of the event effects the sleep-time scheduling. The longer the events, the more often the maximum sleep phase will be scheduled. Also, the possibility that due to the scheduling a event is skipped is reduced with increasing *activity event* length.

5 Evaluation

Five challenges for activity recognition on mobile phones have been identified: **flexibility**, **extensibility**, **robustness**, **resources** and **conditionality**. For many challenges the modular architecture offers solutions, but for other challenges other novel improvements to the activity recognition are proposed. All these challenges and solutions have been discussed and analyzed in detail in the last section. In this section the approaches are experimentally evaluated with real data from mobile phones which reflect the respective challenge.

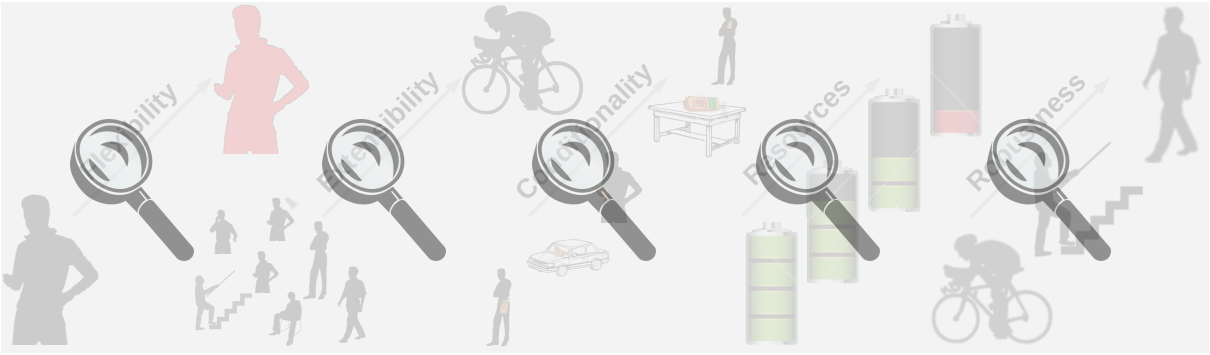


Figure 38: Challenges of activity recognition on mobile phones - evaluation.

First, the adaptability and exchangeability of modules in a dynamic queue to offer **flexibility** is evaluated. In initial experiments the general possibility of adapting classifiers via bit-vector masking to changed conditions is evaluated. The initial evaluation is based on a non modular system without recurrence to eliminate their influence on the results. Finally in the evaluation on the challenge **flexibility** the modular activity recognition is experimentally evaluated in detail on adaptability and exchangeability.

Second, the modular activity recognition is evaluated on the capabilities of being **extended**. A running classifier queue with four modules is **extended** by two additional modules. The **extended** system is evaluated against a modular activity recognition where all six modules are trained together. The experimental evaluation is based again on real data with high complexity and depth from an actual mobile phone.

Third, the improvement due to the recurrent mapping function is evaluated according to the challenge **robustness**. Initial experiments on bases of a system without modularity of the classification and covariant membership functions in the mapping give an general idea of the improvements to the **robustness** due to the recurrence. The overall modular system is then experimentally evaluated in detail on the improvements due to the filtering upon the reliability measure.

Fourth, the accuracy and complexity of a classifier system is evaluated starting from a monolithic and ending with a modular approach to have an initial understanding of the capabilities of a modular approach. The complexity of the activity recognition is influencing the consumption of the **resource** processor and indirectly the **resource** battery. Furthermore, the complexity of a dynamic classifier queue is evaluated in detail by comparing the different possibilities of subdividing the overall recognized classes among the modules. Finally, the processor load and the energy consumption of the modular activity recognition with and without sleep-time scheduling are measured on the OpenMoko Freerunner mobile phone.

Fifth, the modularity is experimentally evaluated on the capabilities to cope with various **conditionalities** of the device and the user. The evaluation of the system on a high amount of *conditional contexts* and activity classes with a varying amount of modules should indicate the possibilities of a modular activity recognition according to **conditionality**. Again, the data base is the accelerometer of an actual mobile phone which is used in these various conditions by three different users.

Last, two tradeoffs are exemplarily evaluated, which deliver actual values for the tradeoff. One tradeoff is between the accuracy of the activity recognition and the activity event detection rate. The filtering upon the reliability measure improves the overall statistical accuracy of the modular activity recognition, but also reduces the amount of classifications passed to the next instance of processing or an application. The reduction of passed classifications

could result in the missing of activity events. The other tradeoff which is experimentally evaluated is between the challenges **robustness** and **resources**. The recurrent mapping function stabilizes the classification process, but in combination with a sleep-time scheduling it reduces the recognition accuracy.

5.1 Evaluation: Challenge 1 - Flexibility

As **flexible** activity recognition was defined, where each module is individually adaptable and exchangeable. It was analyzed how an adaptation of modules can be performed without destroying the original classification capabilities and which can be removed at any time. The exchange of modules sometimes also needs the adaptation of other modules, since they recognize transitions onto the exchanged module. The recognition of these transitions could be faulty with the a new module in the queue.

First, the method of adapting classifiers due to bit-vector masking is evaluated. In the experiments a non recurrent FIS with and without covariant membership functions is used. Also, a monolithic approach is favored over a modular one, since these initial experiments should only evaluate the capabilities due to the bit-vector masking and the covariant membership functions.

Last, the approach of adaptation and exchange of modules is evaluated for a system with three modules. The scenario considers one module in the queue, which is affected through changed conditions. The module is adapted onto the changed conditions with a bit-vector masking. Also, the module is exchanged with a module directly trained on the new conditions. The accuracy of the adaptation and the exchange of this one module is compared to an activity recognition as a whole and directly trained on the new conditions.

5.1.1 Bit-Vector Adaptation

In this subsection the adaptation through the bit-vector masking is analyzed solely without the effects of a modular classifier with recurrent mapping functions. The results presented here had been previously published in [31]. The evaluation is based on a smart artifact and not on a mobile phone, but the results are transferable.

The data for the examples were collected with a typical ubiquitous and pervasive artifact, the *AwarePen* [27, 29]. This artifact consists of a wireless communication device, a digital signal processor, and a 3-dimensional acceleration sensor. It is used to detect several states that a pen can be situated in, e.g. *writing with*, *lying on desk*, *point with*, *in pocket while walking*, etc. The input of the classifier is the variance and the mean over a window length of eight samples for every axis of acceleration, which results in 6 inputs. Here in particular the bit-vector masking is analyzed, so the activity recognition can be adapted to changed conditions.

Example 1: Pen Acceleration Data - 3 Classes In the first example the performance of the bit-vector masking approach for a classifier that has three target classes is shown. These classes are: *lying on desk* (1), *writing* (2), and *pointing at slide* (3). The genetic algorithm's optimization is used to reduce the classification error for data that differs from the training data. Differences in the data for the *lying on desk* is introduced by having a cell phone vibrating next to the pen, thus resulting in slightly different motion data. The training data for the *writing* class was recorded whilst writing on a desk, whereas the unknown data results from writing on a vertical white board. Here, the mean values of the inputs are different, since the orientation of the pen has changed significantly.

Upon the training data set of 900 data pairs, with about 300 per each of the three classes, the system was generated with the previously introduced method. The percentage of correct classifications for this training data set of the resulting covariant TSK-FIS mapping function was 96% and for an independent test set (no unknown data) 94%. Another data set, called check data set, was used to optimize the system with the genetic algorithm. Half of the check data with 900 samples consisted of similar data to the test set and half of it was the previously described unknown data. The classification accuracy before optimization was $\sim 80\%$. After selecting the best result over several runs of the genetic algorithm, the the masked classifiers's accuracy was improved to up to $\sim 99\%$.

To test the optimization a test set (also about 500 samples) was used, different to the previously used sets, which consisted of training set similar and unknown data. The accuracy before optimization was $\sim 85\%$ and $\sim 97\%$ after. The confusion matrices of the test data set for before (table 3, left) and after (table 3, right) optimization show how the different classes have improved, and where the false classifications are situated.

Example 2: Pen Acceleration Data - 5 Classes In the second example, it is shown that the approach also works for more classes and more unrelated check data than in the previous example. This is an upper limit of the adaptation through the bit-vector masking, but still gives reasonable results. The classes are: *lying on desk* (1), *writing* (2), *pointing at slide* (3), *pen in trouser's pocket whilst sitting* (4), and *pen in trouser's pocket whilst standing* (5).

designated classes	classified onto class		
	1	2	3
1	68.72	23.08	8.21
2	0	91.79	8.21
3	0	6.12	93.88

designated classes	classified onto class		
	1	2	3
1	100	0	0
2	0.48	93.24	6.28
2	3.06	0	96.94

Table 3: Confusion matrix for check data (83%, MSE 0.5819) of non adaptated FIS (**left**); Confusion matrix for check data (97%, MSE 0.5819) of adaptated FIS (**right**).

For all these target classes, extra motion patterns were introduced during the test dataset to evaluate for **flexibility** through the bit-vector masking approach. For the *lying on desk* class, the new data contains noisy data from a ringing cellphone next to the pen, and the desk it is lying on being bumped into. The class *writing* is constructed on *writing on desk* data, and is masked also for *writing on white board*. A difference to the *pointing at slide* class compared to the training set is introduced by turning the pen around and pointing with the front end at the slide. The *pen in the trouser's pocket whilst standing* class has unknown data from a different and more active user, than the user in the training data. The last class *pen in the trouser's pocket whilst sitting* is the most challenging one, since the user was once sitting on the edge of the chair, having her legs in a different angle, and another time the user is nervously moving her legs around.

The automatically constructed covariant TSK-FIS classifiers is trained on a data set of 546 data pairs, which are less pairs for more classes than in the previous example. The resulting FIS mapping function was consisting of 18 rules, which was 6 less than the upper bound from the subtractive clustering. The accuracy for the training data set was $\sim 99\%$ and for an independent but similar (no unknown data) check set was $\sim 90\%$. This high classification accuracy for fewer rules shows good coverage through the covariant membership functions (see eqn. 10). Detailed results can be seen for the test set in the confusion matrix of table 4 on the left side. The classification

designated classes	classified onto class				
	1	2	3	4	5
1	89.47	10.53	0	0	0
2	0	98.48	1.52	0	0
3	0	0	100	0	0
4	0	12.5	31.25	56.25	0
5	0	0	0	0	100

designated classes	classified onto class				
	1	2	3	4	5
1	98.23	1.77	0	0	0
2	0	92.31	7.69	0	0
3	7.03	60.94	31.25	0.78	0
4	0	0	0	98.94	1.06
5	0	3.92	19.61	0	76.47

Table 4: Confusion matrix for test data (90%, MSE 0.5819, 273 samples) of constructed FIS without optimization (**left**); Confusion matrix for check data (78%, MSE 3.3464, 554 samples) of adaptated FIS (antecedent+consequence)(**right**).

accuracy of $\sim 54\%$ on the check data for this non-adaptated covariant TSK-FIS mapping function is far from applicable. The genetic optimization - as described previously in subsection 3.3.7 - has therefore to improve the classifier substantially to get reasonable classification results. The bit-vector genome is 108 bits long, which results in an increased search space for the optimization algorithm. After optimization, the accuracy for the check set is 78% and for the test set for adaptation validation $\sim 75\%$. While improving the classification for the new data, the classification accuracy for the old test set (no unknown data) needs to stay the same, or at least worsen just a bit, which is even improved with an accuracy of $\sim 91\%$ (before adaptation $\sim 90\%$). The genetic optimization was done for both, the antecedent and the consequence part of each rule. The confusion matrices for both sets can be found in table 4 on the right and 5 on the left side.

For a bit-vector masking, only on the antecedent part of the rules (see eqn. 11), which is the proposed way in this thesis, the classification accuracy has slightly improved to $\sim 79\%$ (MSE 1.3127) for the check set, $\sim 79\%$ (MSE 5.9081) for the test set, and the test set for construction of the FIS mapping function $\sim 86\%$ (before adaptation $\sim 90\%$). The confusion matrix for this classifier is listed in table 5 on the right side.

designated classes	classified onto class				
	1	2	3	4	5
1	95.45	2.73	1.82	0	0
2	0.91	84.55	10.92	3.64	0
3	9.84	57.38	32.79	0	0
4	0	0	0	98.21	1.79
5	0	4	28	0	68

designated classes	classified onto class				
	1	2	3	4	5
1	88.18	7.27	4.55	0	0
2	2.73	94.55	0.91	1.82	0
3	3.28	68.86	23.77	3.28	0.82
4	0.89	0	1.79	97.32	0
5	0	2	0	0	98

Table 5: Confusion matrix for test data (75%, MSE 4.0262, 554 samples) of adaptated FIS (antecedent+consequence)(**left**); Confusion matrix for test data (79%, MSE 5.9081, 554 sample) of adaptated FIS (antecedent)(**right**).

5.1.2 Adaptation and Exchange of Modules in a Dynamic Queue

In this subsection an extensive evaluation of the **flexibility** approach offered through the adaption or exchange of modules in a dynamic queue is presented. Many combinations of adapted and exchanged modules are compared towards the accuracy of the original classifier queue or a directly trained modular activity recognition. Even though many combinations are evaluated, there are still cases left for analysis, but due to the various possibilities only the more important ones are presented here.

Evaluation Setting A classifier queue is trained on the accelerometer sensor data of two users. The source of the sensor data is the OpenMoko Freerunner phone, which is equipped with two 3-axis accelerometers. Each module recognizes three different activity classes and represents one *conditional context*. The combinations of activity classes, *conditional contexts* and classifier modules are listed in table 6. The sensors are sampled with 100Hz and

Conditional Context	Context Class	Class No.	Classifier Module
Phone in users trouser pocket: *no movement*	user is sitting user is standing user is lying	1 2 3	$\mathbf{M}_1$
Phone in users trouser pocket: *movement*	user is walking user is climbing stairs user is cycling	4 5 6	$\mathbf{M}_2$
Phone in users hand:	just holding talking on phone typing text message	7 8 9	$\mathbf{M}_3$

Table 6: Activity classes, *conditional contexts* and classifier modules for the acceleration sensor.

the feature extraction methods used are mean and variance, which are defined in subsection 3.1.2. The window size of eight measurements leads to 12.5 12-dimensional feature vectors per second. The training data consisted of about 800 and the check data of about 300 data pairs for each class. The training data for the complementary class of module $\mathbf{M}_i$ consisted of 1600 and the check data of 600 data pairs randomly selected out of the training and check data of the other modules $\mathbf{M}_j$ with $i \neq j$. The test data, which was used to evaluate the approach, consisted of about 1700 from train and check data independent data pairs per activity class.

Since the challenge of **flexibility**, which is coped with the modular activity recognition system presented in this thesis, demands the successive handling of unexpected data with pre-trained and existing systems, the test data set is extended with data from a new third user. Only the activity classes classified by module $\mathbf{M}_3$ are represented in this new evaluation data set. This new user provides data, that is very different from the usually expected and trained on sensor data, therefore the recognition accuracy of the module $\mathbf{M}_3$ is significantly lowered. This data is again

separated into training, check and test data, which is used to train or adapt modules and to evaluate the approach to cope with the challenge **flexibility**.

Modular Activity Recognition - Accuracy on Expected Data First, the trained modular classifier is evaluated on similar data on which it was trained on. The test data set which has been used for evaluation consisted of about 1700 data pairs for each activity class and 15650 in total. This data set therefore has the length of over 20 minutes of human activities. The data set had been randomized in slices of 50 data pairs (4s), since this is a stress test for the

	$\mathbf{M}_1$			$\mathbf{M}_2$			$\mathbf{M}_3$		
	1	2	3	4	5	6	7	8	9
1	80.3	11.8	2.2	5.7	6.1	3.2	26.6	2.3	5.6
2	5.1	60.7	3.5	1.1	0.5	0.5	0.4	0.2	4.9
3	0.1	0.2	84.4	0.0	0.0	0.0	0.0	0.0	0.0
4	1.4	11.9	1.2	66.0	24.6	3.9	5.0	0.7	0.9
5	1.2	3.7	1.0	24.4	66.4	4.6	1.2	0.2	0.0
6	0.1	1.2	5.1	0.4	0.4	85.4	0.8	0.0	0.0
7	10.4	5.9	2.5	2.1	1.2	1.8	56.8	6.6	3.0
8	1.0	4.2	0.1	0.3	0.7	0.2	4.6	89.4	11.0
9	0.4	0.4	0.0	0.0	0.0	0.1	0.9	0.6	74.5
	100	100	100	100	100	100	100	100	100

	$\mathbf{M}_1$			$\mathbf{M}_2$			$\mathbf{M}_3$		
	1	2	3	4	5	6	7	8	9
1	91.4	3.8	0.5	6.5	11.3	1.4	27.8	0.3	2.1
2	1.0	84.9	0.1	1.8	1.1	0.0	0.0	0.0	1.0
3	0.0	0.0	99.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.5	2.4	0.1	78.0	8.1	1.4	1.7	0.1	0.4
5	0.0	1.3	0.1	8.9	76.9	1.4	0.6	0.0	0.0
6	0.0	2.8	0.1	1.8	0.0	94.4	0.0	0.0	0.0
7	6.9	1.8	0.1	3.0	2.2	1.4	65.6	1.1	1.9
8	0.0	2.8	0.0	0.0	0.5	0.0	0.6	98.5	5.4
9	0.2	0.4	0.0	0.0	0.0	0.0	3.7	0.0	89.2
	31.7	33.1	78.5	6.1	6.4	32.9	18.9	83.9	21.9

Table 7: Modular classifier system accuracy on expected data for threshold $\tau = 0.0$ (left) with 73.8% and $\tau = 0.9$ (right) with 86.4% overall accuracy.

recurrent mapping function of the modular classification. Why this is a test under extreme conditions is evaluated in subsection 4.3. Also, the filtering upon the reliability measure can improve the overall statistical accuracy, which is again evaluated in subsection 4.3. The confusion matrices for non filtered and filtered classifications upon a threshold $\tau = 0.9$ are listed in table 7. As can be seen, the recognition accuracy without filtering is 73.8% (table 7, left) and with filtering it is 86.4% (table 7, right). Since the filtering reduces the amount of classifications, the last row of the confusion matrices and all following ones the percentage of remaining classifications are listed.

Most classes (5) are recognized with up to or over 90% correct classifications after the filtering, others (3) with an accuracy of over or up to 80%. Only the class no. 7, which identifies the activity *just holding*, is recognized with only 66% accuracy. This is due to the strong interference with the class no. 1 of activity *user is sitting* where 27.8% were falsly classified on. This two activities seem to include feature vectors with similar data, which could indicate a related orientation of the mobile phone for this activities with nearly no movement.

The recognition results indicate, that the modular classification of nine classes with additional filtering on the reliability measure has an accuracy, which is applicable in a real application. Some classes are better distinguishable then others, therefore the system can be improved if some classes are not recognized together in one modular classifier.

Modular Activity Recognition - Accuracy on Unexpected Data The modular activity recognition is facing a challenge in the next step of the **flexibility** evaluation. If the user's behavior changes due to e.g. changed physical or mental condition, the sensor patterns might also be different to the data the modular classifier had been trained on. This is simulated through data of a new user, with which the activity recognition is now confronted. The data of a third user, which has not been present at training, simulates a changed condition for the classes of module $\mathbf{M}_3$, where the user is holding the phone. Since the evaluation test data for only the classes *just holding*, *talking on phone* and *typing text message* is substituted, the classification accuracy for the rest of the activities should not be influenced. The results of the classification are described in the confusion matrices listed in table 8. Here the percentage of correct classifications are significantly lowered for the classes of module $\mathbf{M}_3$, which also lowers the overall accuracy of the activity recognition. The accuracy of the modular classifier without filtering is 59% (table 8, left) and with filtering on threshold $\tau = 0.9$ is 70.8% (table 8, right). Therefore the accuracy is 14.5*pp* (*percentage points*) lowered for no filtering and 15.6*pp* for the filtered modular classification. Especially the accuracy for the

	$\mathbf{M}_1$			$\mathbf{M}_2$			$\mathbf{M}_3$		
	1	2	3	4	5	6	7	8	9
1	85.2	14.8	2.3	5.0	6.4	5.4	78.0	49.4	14.2
2	5.5	65.9	2.9	1.4	0.9	1.1	0.0	16.0	15.7
3	0.1	0.2	90.3	0.0	0.2	0.0	0.0	1.4	0.0
4	1.8	12.0	0.6	64.7	24.4	4.0	1.2	13.6	1.5
5	0.7	3.3	1.2	27.3	67.1	4.5	0.0	4.4	0.1
6	0.0	0.6	0.6	0.9	0.5	83.6	0.0	0.0	0.0
7	5.1	1.0	1.8	0.7	0.3	0.9	18.1	6.8	6.8
8	1.0	2.0	0.3	0.0	0.2	0.1	1.2	8.1	10.7
9	0.6	0.0	0.1	0.0	0.0	0.1	0.0	0.3	50.9
	100	100	100	100	100	100	100	100	100

	$\mathbf{M}_1$			$\mathbf{M}_2$			$\mathbf{M}_3$		
	1	2	3	4	5	6	7	8	9
1	92.7	5.2	0.4	5.1	12.1	2.5	82.0	57.9	10.2
2	1.3	88.7	0.1	2.6	2.6	0.0	0.0	6.3	0.7
3	0.0	0.0	99.2	0.0	0.0	0.0	0.0	0.0	0.0
4	0.6	1.6	0.0	75.0	7.4	1.5	0.0	14.3	0.0
5	0.0	1.6	0.1	12.2	77.9	1.0	0.0	1.6	0.0
6	0.0	1.5	0.1	3.8	0.0	94.4	0.0	0.0	0.0
7	5.1	0.1	0.1	1.3	0.0	0.6	18.0	5.6	3.4
8	0.3	1.2	0.0	0.0	0.0	0.0	0.0	13.5	7.5
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.8	78.2
	34.0	33.3	84.0	5.7	6.6	31.5	14.8	10.9	17.1

Table 8: Modular classifier system accuracy on unexpected data for threshold $\tau = 0.0$ (left) with 59.3% and $\tau = 0.9$ (right) with 70.8% overall accuracy.

classes of $\mathbf{M}_3$ are lowered with 47.9*pp* for the filtered activity recognition. The modular activity recognition system is therefore not able to classify the unexpected data for module $\mathbf{M}_3$, where not the module itself is misclassifying the data, but mostly the wrong module $\mathbf{M}_1$ is falsely classifying it.

Most of the wrong classifications are made on the class no. 1 of activity *user is sitting*. It seems that the new user, whom provides the unexpected data for the phone being held, is holding the phone in such a way it produces a pattern very similar to the phone being in the other users pants pocket and when the users are *sitting*. The class no. 1 was interfering with most of the other classes before, where only expected data was classified, but now the module $\mathbf{M}_1$ responsible for recognizing this class prevents the correct module $\mathbf{M}_3$ from being activated in many cases. Nevertheless, the methods presented here help to make the activity recognition more **flexible** and improve the overall statistical accuracy of the modular classification.

Bit-Vector Adapted (MSE) Modular Activity Recognition - Accuracy on Unexpected Data An adaptation through a bit-vector masking of only the module $\mathbf{M}_3$ which should classify the new unexpected data would not improve the recognition accuracy very much. This is due to the module $\mathbf{M}_1$ which falsely classifies the data which is intended for module $\mathbf{M}_3$. The module $\mathbf{M}_3$ therefore rarely gets activated. Here now, the adaption of all modules via bit-vector masking gets evaluated. The bit-vector for each module is identified through a genetic algorithm search upon the RMSE (Root Mean Squared Error), which has previously been described in subsection 3.3.7.

The search space of the genetic algorithm is determined according to the length of the input which is $n = 13$ (due to the recurrent edge) and the number of rules. The number of rules is individual to every module, where the module $\mathbf{M}_1$ has $m_1 = 8$, $\mathbf{M}_2$ has $m_2 = 8$ and $\mathbf{M}_3$ has $m_3 = 9$ rules. According to this the search space for the bit-vector identification of module $\mathbf{M}_1$ is $2^{13*8} = 2^{104}$, $\mathbf{M}_2$ is 2^{104} and $\mathbf{M}_3$ is 2^{117}. This is a very large search space, but with a genetic algorithm heuristic, a good bit-vector can be found in a reasonable amount of time. Since its an indeterministic heuristic the search time can vary between a few minutes and many hours. Therefore, no estimation can be made about the runtime of the genetic algorithm. Nevertheless, the search space can be limited, but this is clearly part of future work.

The training data set, the genetic algorithm identifies the bit-vectors upon, is a combination of the original training data and the unexpected data provided by the new user. The training data of the modules $\mathbf{M}_1$ and $\mathbf{M}_2$ is nearly the same as used in the original training in spite of the data which should result in the complementary class, where the data of module $\mathbf{M}_3$ was extended with a selection of the unexpected data. For the training data of module $\mathbf{M}_3$ it is vice versa. Here the native classes are only adapted on the new unexpected data and for the complementary class, the original data is used.

The genetic algorithm search is executed three times for each of the three modules. After the search, the best performing bit-vectors are used to mask the modules $\mathbf{M}_1$ to $\mathbf{M}_3$. The results of the bit-vector masked modular classifier are expressed through the confusion matrices for no filtering (left) and filtering upon threshold $\tau = 0.9$ (right) in table 9. The overall recognition accuracy was not improved for the unfiltered classification (table 9, left), in fact the accuracy was lowered by 2,4*pp* to 56.9%. For the filtered modular classification the improvement is

	$\mathbf{M}_1$			$\mathbf{M}_2$			$\mathbf{M}_3$		
	1	2	3	4	5	6	7	8	9
1	76.9	12.2	1.6	5.3	5.8	4.1	69.7	52.2	18.2
2	4.7	65.4	2.6	1.5	1.0	2.5	0.0	14.5	26.3
3	0.1	0.3	87.6	0.0	0.2	0.1	0.0	1.1	0.0
4	2.9	14.7	0.8	53.2	22.4	4.4	0.9	10.7	2.3
5	1.1	1.2	1.4	38.2	69.7	8.5	0.3	3.0	0.0
6	0.0	0.0	0.3	0.5	0.6	78.1	0.0	0.1	0.0
7	14.0	4.8	5.5	1.1	0.3	2.2	28.8	12.0	2.9
8	0.3	0.3	0.1	0.1	0.0	0.0	0.3	6.2	0.9
9	0.0	1.0	0.1	0.0	0.0	0.0	0.0	0.0	46.4
	100	100	100	100	100	100	100	100	100

	$\mathbf{M}_1$			$\mathbf{M}_2$			$\mathbf{M}_3$		
	1	2	3	4	5	6	7	8	9
1	89.1	4.2	0.1	11.2	14.2	1.7	38.7	57.3	11.9
2	1.2	93.7	0.2	4.5	4.5	0.2	0.0	4.9	8.3
3	0.0	0.0	97.6	0.0	0.0	0.0	0.0	0.0	0.0
4	0.3	0.4	0.3	50.6	11.9	0.2	0.0	11.9	1.8
5	0.0	0.3	0.1	27.0	67.9	2.2	0.0	1.4	0.0
6	0.0	0.0	0.1	2.2	0.7	92.3	0.0	0.0	0.0
7	9.3	1.0	1.6	3.4	0.7	3.2	61.3	8.4	7.3
8	0.2	0.0	0.0	1.1	0.0	0.0	0.0	16.1	0.0
9	0.0	0.4	0.0	0.0	0.0	0.0	0.0	0.0	70.6
	31.9	30.3	82.5	3.2	4.6	26.5	36.8	12.4	12.7

Table 9: Bit-vector adapted (MSE) modular classifier system accuracy on unexpected data for threshold $\tau = 0.0$ (left) with 56.9% and $\tau = 0.9$ (right) with 71.0% overall accuracy.

marginal by only 0.2*pp* up to 71%. Only the amount of correct classifications of module $\mathbf{M}_3$ was improved by 12.7*pp* up to 49.2% for the filtered classifier. Therefore, the recognition accuracy for the classes no. 1 to 6 must have been lowered, which can be seen in table 9 on the right side. So the improvment of the recognition rates of module $\mathbf{M}_3$ was done on cost of the other modules $\mathbf{M}_1$ and $\mathbf{M}_2$. Since the identification of the bit-vectors was done on the RMSE of the RFIS mapping on a training data set, the other metric of percentage of correct classified feature vectors could result in a better classification.

Bit-Vector Adapted (%) Modular Activity Recognition - Accuracy on Unexpected Data An actual improvement was reached when identifying the bit-vectors upon the fitness of percentage of correct classifications. The confusion matrices again for no filtering (left) and filtering on threshold $\tau = 0.9$ (right) are listed in table 10. It was expected, that the accuracy for an unfiltered classification would improve, but also the filtered accuracy is higher than with an identification upon the RMSE. This results contradict the assumed behavior, where the optimization on the RMSE should result in a more expressive reliability measure. The accuracy of the non filtered recognition had been improved by 7.2*pp* up to 66.5% (table 10, left) and the filtered by 5,7*pp* to 76.5% (table 10, right). Nevertheless, the accuracy of the classes the unexpected data regards are improved by 42.5*pp* to 79.1%, but the

	$\mathbf{M}_1$			$\mathbf{M}_2$			$\mathbf{M}_3$		
	1	2	3	4	5	6	7	8	9
1	72.5	11.2	1.3	4.7	5.3	7.5	14.8	11.5	6.2
2	0.5	70.4	0.5	1.2	2.1	0.9	0.0	32.9	0.1
3	0.0	0.4	82.8	0.2	0.4	0.1	0.0	4.7	0.0
4	2.5	4.9	1.1	68.1	23.5	2.8	0.0	14.5	1.4
5	2.4	2.2	2.1	22.7	65.3	3.7	0.0	7.7	0.6
6	0.1	0.1	0.4	0.7	1.8	63.3	0.0	0.3	0.0
7	17.6	4.7	11.5	1.8	0.9	19.9	84.3	14.2	1.3
8	2.8	1.9	0.2	0.4	0.6	0.5	0.9	13.6	1.0
9	1.3	4.2	0.0	0.0	0.0	0.1	0.0	0.3	78.1
	100	100	100	100	100	100	100	100	100

	$\mathbf{M}_1$			$\mathbf{M}_2$			$\mathbf{M}_3$		
	1	2	3	4	5	6	7	8	9
1	74.9	6.7	0.3	3.8	9.0	3.2	2.0	7.9	5.3
2	0.5	77.3	0.2	1.3	3.2	0.2	0.0	26.6	0.0
3	0.0	0.0	79.7	0.0	0.5	0.0	0.0	1.4	0.0
4	0.3	1.3	0.7	80.7	11.2	0.5	0.0	6.5	0.0
5	0.5	2.7	0.3	12.2	71.8	0.9	0.0	0.0	0.0
6	0.0	0.0	0.2	1.3	1.6	66.6	0.0	0.0	0.0
7	22.7	4.7	18.6	0.4	2.1	28.3	98.0	10.1	2.4
8	0.8	2.0	0.0	0.4	0.5	0.0	0.0	47.5	0.5
9	0.3	5.3	0.0	0.0	0.0	0.2	0.0	0.0	91.7
	20.2	6.2	32.5	8.6	6.5	28.3	73.6	12.0	24.0

Table 10: Bit-vector adapted (%) modular classifier system accuracy on unexpected data for threshold $\tau = 0.0$ (left) with 66.5% and $\tau = 0.9$ (right) with 76.5% overall accuracy.

classification accuracy of the other classes (class no. 1-6) was again lowered by 12.8*pp*. This is a tradeoff where the importance of classifying all classes with a reasnoable or just a subset with very high accuracy needs to be weighted. In fact, the recognition results of the first six classes of the bit-vector masked modules need to be compared to the accuracy of the original classifier queue on the unexpected, but on the expected sensor patterns. Since the original

unmasked classifier queue can not classify the unexpected data, the classifier can be considered to only recognize six classes. Since a classifier recognizing only six instead of nine classes will always be better, the accuracy on the original sensor patterns need to be the point of comparison. Compared to the original performance the accuracy of recognizing the first six classes was only lowered by 12.2*pp*.

Bit-Vector Adapted (%) Modular Activity Recognition - Accuracy on Expected Data The recognition accuracy for the unexpected data could have been improved to reasonable results, but how does the bit-vector adapted modular classifier perform on the original test data without unexpected data. This question is evaluated here, where the resulting confusion matrices can be found in table 11. As can be seen, the recognition accuracy was significantly

	M_1			M_2			M_3		
	1	2	3	4	5	6	7	8	9
1	73.2	12.4	0.5	5.5	5.8	6.0	43.2	42.2	16.2
2	0.7	71.0	0.3	1.3	2.7	0.8	29.9	29.5	0.1
3	0.0	0.3	82.3	0.2	0.6	0.1	10.1	0.1	0.0
4	1.8	5.2	2.2	68.4	22.6	2.9	2.0	7.2	5.5
5	1.2	2.9	3.2	20.5	64.1	3.8	1.4	0.1	4.4
6	0.0	0.2	0.4	0.7	1.7	61.1	0.2	0.0	0.0
7	18.8	3.2	10.8	2.7	2.0	23.3	9.4	2.3	3.2
8	2.6	1.1	0.2	0.5	0.6	0.7	2.5	14.1	18.2
9	1.4	3.6	0.0	0.0	0.0	0.1	1.0	4.4	48.6
	100	100	100	100	100	100	100	100	100

	M_1			M_2			M_3		
	1	2	3	4	5	6	7	8	9
1	77.5	9.2	0.2	4.7	5.7	1.9	68.3	5.1	3.8
2	0.8	77.0	0.0	1.7	5.2	0.0	5.5	4.8	0.0
3	0.0	0.0	81.2	0.4	1.0	0.0	3.0	0.0	0.0
4	0.3	2.6	0.7	77.9	10.9	0.5	0.0	4.8	1.3
5	0.8	2.6	0.3	12.8	71.9	1.0	1.0	0.0	0.4
6	0.0	0.0	0.2	0.9	1.0	63.7	0.5	0.0	0.0
7	19.6	2.6	17.4	1.3	3.6	32.8	16.6	3.8	18.2
8	0.8	1.3	0.0	0.4	0.5	0.0	0.0	79.0	2.1
9	0.3	4.6	0.0	0.0	0.0	0.2	5.0	2.5	74.2
	20.3	6.3	31.9	8.5	6.6	27.7	10.7	11.7	10.7

Table 11: Bit-vector adapted (%) modular classifier system accuracy on expected data for threshold $\tau = 0.0$ (left) with 54.7% and $\tau = 0.9$ (right) with 68.8% overall accuracy.

lowered by 19.1*pp* to 54.7% for the unfiltered (table 11, left) and 17,6*pp* to 68% for the filtered classifications (table 11, right) compared to the originial classifier queue without bit-vector masking. But if a closer look is taken one can see that some of the activity classes are still recognized with reasonable accuracy after the filtering. Only the the class no. 7 is with 16.6% correct classified feature fectors far from usable in a real application. But the classification for this class had been badly performing before with the original activity recognition system, where it also had the worst accuracy. In this manner, the bit-vector masking only amplified the effect for this class, which had already existed before.

Nevertheless, also the bit-vector adapted classifier modules are able to process the original expected sensor patterns with a reasonable accuracy. This is a strong argument for a bit-vector masking approach and shows the expected, that the masking does not destroy the original classification capabilities. Also, if the data, where the bit-vector is identified on, is not only the data the respective module needs to be adapted on, but also includes some of the old expected patterns, the capabilities of classifying the these would even be better. This is again a tradeoff, since this would also lower the classification accuracy for the new unexpected patterns.

Partly Bit-Vector Adapted (%) Modular Activity Recognition - Accuracy on Unexpected Data Since the most incorrect classifications are made by the module $\mathbf{M}_1$ of the unexpected data and the module $\mathbf{M}_2$ has nearly no influence on the accuracy of module $\mathbf{M}_3$, a masking of only the modules $\mathbf{M}_1$ and $\mathbf{M}_3$ is evaluated. The resulting confusion matrices of this evaluation setting are shown in table 11. The accuracy toward the classifier queue where all modules are masked was only lowered by 0.7*pp* for the unfiltered (table 11, left) and 0.9*pp* for the filtered classification (table 11, right). In both cases the classification accuracy is insignificantly lowered, which confirms the assumption.

Since the identification of a bit-vector needs time, this method could offer a more efficient adaptation of the modular activity recognition. An algorithm would need to check the confusion matrix of the unexpected sens or patterns according to certain rules. Therefore the modules which influence the classification accuracy can be identified and only those would be adapted through a bit-vector masking. Non-modular classifier approaches would not offer such a **flexibility**.

	M_1			M_2			M_3		
	1	2	3	4	5	6	7	8	9
1	70.9	8.6	1.2	2.5	3.0	2.6	18.1	10.1	7.9
2	0.5	55.8	0.8	1.4	1.4	0.7	0.0	27.1	0.1
3	0.0	0.5	80.2	0.2	0.2	0.0	0.0	3.3	0.0
4	3.7	17.3	1.8	65.6	25.1	4.2	0.9	26.5	4.1
5	2.7	5.6	2.2	28.4	69.5	4.9	0.9	9.3	1.2
6	0.2	0.6	0.6	0.9	0.6	84.6	0.0	0.0	0.0
7	17.8	3.8	12.9	0.8	0.2	2.9	79.5	11.2	1.3
8	3.1	3.4	0.2	0.1	0.0	0.1	0.6	11.7	0.9
9	0.9	4.4	0.0	0.0	0.0	0.1	0.0	0.3	74.1
	100	100	100	100	100	100	100	100	100

	M_1			M_2			M_3		
	1	2	3	4	5	6	7	8	9
1	72.5	6.0	0.3	4.6	7.3	0.8	2.9	4.9	7.5
2	0.3	55.4	0.2	2.0	1.6	0.2	0.0	23.6	0.0
3	0.0	0.0	79.4	0.0	0.5	0.0	0.0	1.4	0.0
4	0.8	14.9	0.0	75.8	7.3	1.4	0.0	18.8	0.0
5	0.8	7.7	0.3	13.1	83.4	1.0	0.0	5.6	0.0
6	0.0	7.1	0.3	3.9	0.0	91.3	0.0	0.0	0.0
7	24.8	3.6	19.4	0.7	0.0	5.3	97.1	9.0	2.7
8	0.3	1.8	0.0	0.0	0.0	0.0	0.0	36.8	1.1
9	0.5	3.6	0.0	0.0	0.0	0.0	0.0	0.0	88.7
	19.9	7.0	32.3	5.5	6.7	33.2	71.5	12.5	21.7

Table 12: Partly bit-vector adapted (%) modular classifier system accuracy on unexpected data for threshold $\tau = 0.0$ (left) with 65.8% and $\tau = 0.9$ (right) with 75.6% overall accuracy.

Exchanged Module and Partly Bit-Vector Adapted (%) Modular Activity Recognition - Accuracy on Unexpected Data Until now the modules were only manipulated by different combinations of bit-vector masking, but how does the modular activity recognition perform when the module $\mathbf{M}_3$ is exchanged with a module that is directly trained on both, the expected and unexpected sensor patterns. The data the new module is trained on consists of one third of the new unexpected sensor patterns and two thirds of the old ones. This is because of the amount of users which provide the training data, since two users provided the original and one the new data.

The modules $\mathbf{M}_1$ and $\mathbf{M}_2$ which do not get exchanged need to be masked with bit-vectors. As explained before, the recognition of the complementary class and therefore the transition onto the exchanged module is also effected through the changed sensor patterns. Previous evaluation results have shown, that the module $\mathbf{M}_2$ does not interfere with the new unexpected sensor patterns, so it would not necessarily need to be masked. This is not evaluated, since similar results as before are expected.

The resulting confusion matrices for the same evaluation test set of unexpected patterns as before are shown in table 13. For the non filtered classifier queue the confusion matrix is listed on the left and for the filtered on the right. This approach offers the best performance so far. The accuracy for the non filtered could be increased

	M_1			M_2			M_3		
	1	2	3	4	5	6	7	8	9
1	62.4	9.1	0.8	3.5	4.0	8.1	9.2	6.4	1.4
2	0.5	70.3	0.5	1.1	2.0	1.0	0.0	12.4	0.1
3	0.0	0.3	84.5	0.2	0.4	0.1	0.0	1.6	0.0
4	1.3	4.9	1.0	67.9	23.1	3.1	0.0	6.5	0.7
5	1.7	2.1	0.6	22.7	66.2	4.5	0.0	4.1	0.2
6	0.1	0.1	0.4	0.7	1.8	75.4	0.0	0.2	0.0
7	29.6	5.2	6.5	2.5	1.8	3.7	89.6	22.3	1.0
8	2.8	7.2	5.6	1.2	0.7	2.8	1.2	46.3	11.5
9	1.4	0.8	0.0	0.0	0.0	0.0	0.0	0.2	84.8
	100	100	100	100	100	100	100	100	100

	M_1			M_2			M_3		
	1	2	3	4	5	6	7	8	9
1	83.6	7.6	0.2	2.6	6.7	3.8	3.7	1.9	0.0
2	0.9	70.7	0.2	1.3	3.1	0.3	0.0	13.0	0.0
3	0.0	0.0	95.0	0.0	0.0	0.0	0.0	0.9	0.0
4	0.0	1.3	0.8	80.2	11.3	0.6	0.0	4.6	0.0
5	0.9	1.9	0.4	13.4	70.6	1.5	0.0	0.0	0.0
6	0.0	0.0	0.2	1.3	1.5	91.3	0.0	0.0	0.0
7	7.6	7.0	1.8	0.9	2.6	1.5	96.3	15.7	0.0
8	7.1	10.2	1.4	0.4	4.1	1.2	0.0	63.9	1.7
9	0.0	1.3	0.0	0.0	0.0	0.0	0.0	0.0	98.3
	12.1	6.5	26.8	8.4	6.7	22.5	24.0	9.4	48.7

Table 13: Modular classifier system with exchanged module $\mathbf{M}_3$ which was directly directly trained on expected and unexpected data and tested on unexpected data for threshold $\tau = 0.0$ (left) with 71.9% and $\tau = 0.9$ (right) with 83.3% overall accuracy.

by 5.4*pp* and the filtered by 6.8*pp* compared to the approach where all modules were masked with bit-vectors. Many of the activity classes are now classified with above 90% accuracy for the filtered classifier queue. Only one class, the class no. 8 of activity *talking on phone*, is classified with below 70% correct classifications. This class strongly interferes with the class no. 2 of activity *standing* and class no. 7 *just holding*. A majority of the incorrect

classifications are mutually misclassified by this three activities. Nevertheless, this evaluation results show, that the exchange of classifier modules can improve the **flexibility** of a activity recognition.

Exchanged Module and Partly Bit-Vector Adapted (%) Modular Activity Recognition - Accuracy on Expected Data The exchanged module $\mathbf{M}_3$ was performing well on the unexpected sensor patterns, but how is the accuracy on the expected data. The results of the evaluation of the classifier queue with exchanged module $\mathbf{M}_3$ are again listed in confusion matrices in table 14. The approach with exchanged module $\mathbf{M}_3$, filtered classification

	$\mathbf{M}_1$			$\mathbf{M}_2$			$\mathbf{M}_3$		
	1	2	3	4	5	6	7	8	9
1	67.9	9.4	0.5	4.9	4.3	7.2	20.0	25.4	5.9
2	0.6	69.0	0.7	0.9	2.0	0.9	12.3	17.9	0.0
3	0.0	0.2	85.7	0.1	0.4	0.0	4.2	0.0	0.0
4	1.2	4.6	1.1	67.9	22.1	3.2	1.1	6.7	2.0
5	1.2	2.3	1.6	20.9	65.6	4.3	0.8	0.0	2.2
6	0.1	0.2	0.4	0.8	1.8	76.6	0.2	0.0	0.0
7	24.9	3.9	5.6	3.0	2.7	3.3	43.7	1.2	1.1
8	3.1	9.7	4.4	1.4	1.1	3.4	17.7	47.7	24.4
9	0.9	0.7	0.0	0.0	0.0	0.0	0.0	1.0	64.3
	100	100	100	100	100	100	100	100	100

	$\mathbf{M}_1$			$\mathbf{M}_2$			$\mathbf{M}_3$		
	1	2	3	4	5	6	7	8	9
1	84.3	7.9	0.2	3.9	5.5	2.1	25.2	1.8	0.9
2	0.7	70.9	0.0	1.7	3.0	0.0	3.6	2.0	0.2
3	0.0	0.0	96.2	0.4	1.0	0.0	1.2	0.0	0.0
4	0.0	1.8	0.8	78.5	9.5	0.6	0.0	2.0	0.4
5	1.0	1.8	0.4	12.4	70.1	1.5	0.0	0.0	0.0
6	0.0	0.0	0.2	0.9	1.0	92.3	0.0	0.0	0.0
7	7.3	7.9	0.8	1.3	5.0	1.2	50.8	0.4	0.4
8	6.3	9.1	1.4	0.9	5.0	2.4	19.2	94.0	12.4
9	0.3	0.6	0.0	0.0	0.0	0.0	0.0	0.0	85.6
	15.4	6.9	26.9	8.4	6.9	22.1	13.4	21.0	20.8

Table 14: Modular classifier system with exchanged module $\mathbf{M}_3$ which was directly trained on expected and unexpected data and tested on expected data for threshold $\tau = 0.0$ (left) with 65.4% and $\tau = 0.9$ (right) with 80.3% overall accuracy.

output, masked modules $\mathbf{M}_1$ and $\mathbf{M}_2$ is classifying with only 6.1*pp* less accurate than the original classifier queue. The difference between the non filtered classifications is with 8.4*pp* marginally higher, but is still the best tradeoff between classification accuracy on expected or unexpected sensor patterns. The method of exchanging modules and adapting others through bit vector masking is therefore the method of choice, when new unexpected sensor patterns occur due to changed conditions. A final conclusion can be made, when these results are compared to a classifier queue, which was directly and all together trained on the expected and unexpected sensor patterns.

Modular Activity Recognition directly Trained on Unexpected and Expected Data - Accuracy on Unexpected and Expected Data Since the original system was only trained on the expected data, the resulting system was unable to classify the unexpected patterns. But how does a classifier queue perform, which was trained on both, the expected and unexpected sensor patterns? Does this system outperform the classifier queue where the module $\mathbf{M}_3$ was exchanged? Is this system as good as the original classifier queue on the expected patterns? These questions will be answered in this last evaluation scenario of this subsection.

Again the combined data of unexpected and expected sensor patterns was divided in three parts, training, check and test data. The training data had 800 feature vectors per class, the check data 400 and for the complementary class the double amount for each. The rest of the data was used for evaluation, where the expected as well as the unexpected test data had about 1700 data pairs. The classifier queue is evaluated on expected and unexpected patterns separately to have comparable results.

The results of this classifier queue for the test data with unexpected sensor values are shown in table 15. The accuracy on the unexpected test data is with 76.9% after filtering 6.4*pp* lower than the approach with the exchanged module and only 0.4*pp* higher as with the method where all modules were bit-vector masked.

The confusion matrices of the classifier queue for the expected evaluation data are shown in table 16. For this the accuracy has lowered by 0.9*pp* towards the accuracy of the original classifier queue and has increased by 5.1*pp* compared to the queue with exchanged and masked modules.

This results show, that the direct training of all classifier modules on the original and changed conditions does not outperform the approaches of adaptation and exchange. A reason for this could be, that with sensor patterns of three different users instead of two to train the classifier modules onto, the variance is too high to consider all differences

	$\mathbf{M}_1$			$\mathbf{M}_2$			$\mathbf{M}_3$		
	1	2	3	4	5	6	7	8	9
1	76.1	24.3	19.0	4.6	4.9	5.4	48.7	14.5	5.1
2	2.7	30.9	1.3	0.1	0.2	0.2	0.6	2.3	0.1
3	0.3	0.2	51.1	0.1	0.0	0.0	0.0	0.2	0.0
4	5.4	16.9	3.9	65.4	18.6	5.7	0.6	17.8	1.3
5	1.0	9.7	0.9	26.4	71.9	5.5	6.5	5.1	0.1
6	0.0	4.1	2.1	1.1	1.3	81.7	0.0	0.7	0.0
7	12.2	2.7	10.3	2.1	2.7	0.5	40.7	39.8	1.5
8	1.5	0.7	0.9	0.1	0.3	0.6	3.0	18.4	10.0
9	0.7	0.7	0.0	0.0	0.0	0.0	0.0	0.2	81.8
	100	100	100	100	100	100	100	100	100

	$\mathbf{M}_1$			$\mathbf{M}_2$			$\mathbf{M}_3$		
	1	2	3	4	5	6	7	8	9
1	78.5	12.6	1.1	0.8	1.3	0.4	20.9	8.9	1.0
2	0.4	74.0	0.2	0.2	0.3	0.0	0.0	3.6	0.0
3	0.0	0.2	93.0	0.0	0.0	0.0	0.0	0.0	0.0
4	2.6	10.3	1.5	83.2	10.0	0.4	0.0	13.4	0.0
5	0.4	1.5	0.1	14.5	84.9	1.2	18.6	5.4	0.0
6	0.0	0.0	0.0	0.6	0.8	98.0	0.0	3.6	0.0
7	14.8	0.6	4.1	0.6	2.8	0.0	60.5	42.0	0.0
8	1.9	0.2	0.0	0.0	0.0	0.0	0.0	23.2	2.3
9	1.5	0.6	0.0	0.0	0.0	0.0	0.0	0.0	96.6
	14.5	27.1	50.2	17.2	13.5	32.5	12.8	9.7	44.6

Table 15: Modular classifier system directly trained on expected and unexpected data and tested on unexpected data for threshold $\tau = 0.0$ (left) with 57.5% and $\tau = 0.9$ (right) with 76.9% overall accuracy.

	$\mathbf{M}_1$			$\mathbf{M}_2$			$\mathbf{M}_3$		
	1	2	3	4	5	6	7	8	9
1	72.4	23.4	18.0	5.0	6.6	6.1	17.1	39.4	21.1
2	2.8	28.2	1.5	0.4	0.2	0.2	0.6	0.4	1.4
3	0.3	0.2	45.8	0.1	0.0	0.0	0.0	0.3	0.1
4	6.0	16.7	4.9	65.4	18.6	5.8	6.3	1.0	10.2
5	0.8	7.9	0.5	23.8	70.7	5.0	1.7	0.0	0.1
6	0.3	5.7	5.6	0.7	1.1	81.0	0.0	0.1	0.0
7	13.5	4.9	9.9	4.0	2.6	1.4	70.8	1.8	1.3
8	1.6	0.9	1.0	0.6	0.2	0.4	3.2	56.9	13.7
9	1.2	1.0	0.0	0.0	0.0	0.0	0.0	0.1	52.0
	100	100	100	100	100	100	100	100	100

	$\mathbf{M}_1$			$\mathbf{M}_2$			$\mathbf{M}_3$		
	1	2	3	4	5	6	7	8	9
1	79.5	11.5	1.3	1.9	1.8	0.4	4.9	5.2	9.5
2	1.2	74.6	0.5	0.0	0.0	0.0	0.0	0.0	1.0
3	0.0	0.2	91.6	0.2	0.0	0.0	0.0	0.0	0.0
4	1.5	9.4	1.9	84.1	9.0	0.6	3.0	0.4	1.2
5	0.0	1.5	0.1	11.4	86.3	1.0	4.0	0.0	0.0
6	0.4	0.0	0.0	0.0	0.8	97.8	0.0	0.0	0.0
7	12.7	1.3	4.4	2.3	2.1	0.2	87.8	0.7	0.6
8	3.1	0.7	0.0	0.0	0.0	0.0	0.3	93.5	14.8
9	1.5	0.7	0.0	0.0	0.0	0.0	0.0	0.1	72.8
	13.9	25.2	45.1	17.1	13.4	33.1	17.7	27.1	22.0

Table 16: Modular classifier system directly trained on expected and unexpected data and tested on expected data for threshold $\tau = 0.0$ (left) with 60.4% and $\tau = 0.9$ (right) with 85.4% overall accuracy.

equally. A indication of this can be found in the confusion matrices of table 15. E.g. for the classification of the unexpected sensor patterns class no. 8 of activity *talking on phone* is classified with unacceptably low accuracy. Only 23.2% of the feature vectors were classified on the correct class no. 8. Most of the incorrect classifications were made on the class no. 7 of activity *just holding*. Also the classification of class no. 7 was badly performing, since only 60.5% were correct classified. The incorrect classifications of this class were made on the class no. 5 and 1.

The performance on the expected sensor patterns is with 85.4% overall correct classifications much higher. This ratio could be switched in favour of the classification of the unexpected patterns, if the amount of unexpected data in the training data is increased. But since the approach is outperformed in some numbers by the exchange of only module $\mathbf{M}_3$ and adaptation of the modules $\mathbf{M}_1$ and $\mathbf{M}_2$, which also offers more **flexibility** and needs less resources for training, this method does not need to be considered.

Summary and Discussion Since a lot of numbers and results have been presented in this subsection a summary is presented now. Also, the results are discussed in total and how they effect the challenge **flexibility**.

All classification accuracies of the different methods are listed together in table 17 along with the specification of the scenario. The strongest method for classifying both, the expected and the unexpected sensor data patterns is the method where module $\mathbf{M}_3$ is exchanged and modules $\mathbf{M}_1$ and $\mathbf{M}_2$ were masked with bit-vectors. The original classifier was performing best on the data which did not include any unexpected sensor patterns, but this was anyway assumed to be the upper limit.

Thew methods using only adaptation of the modules through bit-vector masking did also perform well, where

Evaluation Description	**expected (e) or unexpected (u) evaluation data**	**Exchanged (e), Adapted (a), Trained (t) or Nothing Done (-)**			**Accuracy on Filter Threshold**	
		$\mathbf{M}_1$	$\mathbf{M}_2$	$\mathbf{M}_3$	$\tau = 0.0$	$\tau = 0.9$
Original modular classifier	e	-	-	-	73.8%	86.4%
Original modular classifier	u	-	-	-	59.3%	70.8%
Bit-vector adapted (MSE)	u	a	a	a	56.9%	71.0%
Bit-vector adapted (%)	u	a	a	a	66.5%	76.5%
Bit-vector adapted (%)	e	a	a	a	54.7%	68.8%
Partly bit-vector adapted (%)	u	a	-	a	65.8%	75.6%
Exchanged mod., partly bit-vec. adap. (%)	u	a	a	e	71.9%	83.3%
Exchanged mod., partly bit-vec. adap. (%)	e	a	a	e	65.4%	80.3%
Directly trained on exp. and unexp. data	u	t	t	t	57.5%	76.9%
Directly trained on exp. and unexp. data	e	t	t	t	60.4%	85.4%

Table 17: Summary of classification results for the different methods for providing **flexibility**.

the combination of only adapting module $\mathbf{M}_1$ and $\mathbf{M}_3$ was only marginally worse than adapting all modules. The effort to identify a bit-vector can not be numeralized, since a genetic algorithm is used which is an indeterministic heuristic. Nevertheless, a limitation of the search space could speed up the search, but this is part of future work.

The direct training of the whole queue of classifier modules on the expected and unexpected patterns is only marginally lower in accuracy than the original queue for the expected sensor data. The performance of this classifier queue is worse than the approach of exchange though. Since this method demands the training of all classifier modules, a increased effort for training has to be considered. Also, the newly training of classifiers when changed conditions occur is possible with monolithic classifiers, too. This approach does not offer true **flexibility** as defined in this thesis.

5.2 Evaluation: Challenge 2 - Extensibility

An **extensible** activity recognition was declared as a classifier, where new parts can be added so more activities can be recognized. This possibly could be done with a monolithic classifier too, although it is very unlikely. With the modular activity recognition presented in this thesis, the addition of new modules recognizing new activities is easily possible. Since transitions between modules are detected due to the recognition of the *complementary class*, the old modules in the dynamic queue need to be bit-vector masked. After being masked, they recognize transitions onto new modules.

In this subsection the **extensibility** of the modular activity recognition is experimentally evaluated according to an example. The addition of new modules to a preexisting queue is compared to a queue of classifier modules, where all modules are trained together. The results will show, that the modular activity recognition is **extensible** with only a marginal loss in classification accuracy.

5.2.1 Evaluation

For the evaluation of the extensibility the accelerometer data of a mobile phone is used. The results presented here were already published in [26]. The phone is required to be with the user, e.g. in her pocket or held in her hand. With the novel modular activity recognition approach, not only activity classes can be recognized, but also the place where the phone is positioned. This is called *conditional context* and directly through the respective activated classifier module can be reacted on it. Initially four modules are used in a dynamic queue to detect ten activity classes and to cope with four *conditional contexts*. To this queue two new modules are added, which recognize five additional activities and two more *conditional contexts*.

Evaluation Setting The device chosen is the 'OpenMoko Freerunner' phone which comes with two 3-D accelerometer sensors. This phone was chosen, since it provides a open source rapid prototyping platform where all interfaces are easily accessible. Also, the two included accelerometer sensors with high accuracy and sampling rates provide an upper bound, which due to the rapid development of the commercially available mobile phones will be reached in the near future. The sensors have a sampling rate of about 100Hz. With a window size of 8 samples 12.5 feature vectors need to be classified per second. The features are mean and variance, as described in subsection 3.1.2. With this feature extraction, the resulting feature vector has 12 dimensions. Adding the recursive dimension makes a total of 13 dimensions. This feature vector is mapped onto the respective activity class via the currently *active* module. If the active module classifies onto the complementary class, the next module is activated for this feature vector. This procedure is repeated for this feature vector until one of the modules classifies onto a class different from the complementary one. This classifier module is then put first in queue and the next feature vector is processed. In case all classifiers including the last one classify onto $\overline{c_i}$, the overall output is the complementary class. This means the feature vector cannot be classified correctly by the given queue. To increase reliability – and thereby accuracy – filtering upon the uncertainty is done. The filtering, which was described in subsection 3.1.5, reduces the amount of output classes, but for most applications a few classes per second are more than enough, where the reliability is most important. Here it is assumed that most activities are not changing faster than in seconds. The correlation and trade off between accuracy and event detection rate (definition follows) is analyzed in detail in subsection 4.6.1 and evaluated in subsection 5.6.1.

The original modular classifier queue classifies onto ten classes and recognizes four *conditional contexts*. To evaluate the addition of new classifiers, two more modules are put in queue. These recognize five classes and two *conditional contexts*. All activity classes, conditional contexts and respective classifier modules are shown in table 18.

The training data $\mathcal{V}_i^{tr}$ for classifier module $\mathbf{M}_i$ consisted of 400 data pairs and the check data $\mathcal{V}_i^{ck}$ of 200 pairs for each class c_{ij}. The data for training the complementary class $\overline{c}_i$ consisted of 800 pairs and for the check data of 400 pairs, which was randomly selected out of the training and check data for the other classifiers. Both sets $\mathcal{V}_i^{tr}$ and $\mathcal{V}_i^{ck}$ were randomized in slices of 30 feature vector pairs, so the recurrence could be trained and tested. Each classifier modules' $\mathbf{M}_i$ RFIS mapping function $\mathbf{S}_i$ was trained for 10 epochs (outer loop), after which the RFIS achieving the best classification accuracy for combination of training and check data was chosen.

The evaluation data $\mathcal{V}^{ev}$ had about 2000 data pairs per activity class, so 160 seconds per activity. This evaluation set was also randomized according to slices of 20 data pairs ($1,67$ seconds). Since most of the false classifications

Conditional Context	Context Class	Class No.	Classifier Module
Phone in users trouser pocket: *no movement*	user is sitting user is standing user is lying	1 2 3	$\mathbf{M}_1$
Phone in users trouser pocket: *movement*	user is walking user is climbing stairs user is cycling	4 5 6	$\mathbf{M}_2$
Phone on table:	no movement	7	$\mathbf{M}_3$
Phone in users hand:	just holding talking on phone typing text message	8 9 10	$\mathbf{M}_4$
Phone in users trouser pocket:	user is sitting in bus user is standing in bus	11 12	$\mathbf{M}_5$
Phone in users trouser pocket: *dancing*	user is dancing (style 1) user is dancing (style 2) user is dancing (style 3)	13 14 15	$\mathbf{M}_6$

Table 18: Activity classes, *conditional contexts* and classifier modules for the acceleration sensor. Additional new modules are shaded grey.

are occurring when changing from one activity class to another (due to recurrence), this is a stress test for the classifier modules. In a real application it is estimated that even better results can be reached than presented in the following, because activities change less often.

Evaluation Results The first evaluation results presented here are from a modular activity recognition, where all classifier modules get trained on a collective data set. When no filtering upon the uncertainty value is done (see tab. 19), the overall recognition rate lies at only 58%. This is too low for practical use, but with the Recurrent Fuzzy Inferance System (RFIS) approach, the results can be improved significantly. As mentioned before, activity data from sources without fixed position is hardly separable. For this a filtering is needed, where the classifications with low reliability are separated from the recognitions with high reliability.

	$\mathbf{M}_1$			$\mathbf{M}_2$			$\mathbf{M}_3$	$\mathbf{M}_4$			$\mathbf{M}_5$		$\mathbf{M}_6$		
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	86.1	2.6	7.8	1.3	2.8	13.2	5.6	3.3	0.9	12.3	19.7	21.6	1.7	3.9	6.7
2	0.7	58.2	3.4	0.1	0.3	7.9	0.4	1.1	1.4	1.4	11.4	1.7	0.7	0.8	2.9
3	0.0	0.4	76.0	0.0	0.4	4.1	0.0	0.0	1.3	0.0	0.0	0.1	0.3	0.1	0.7
4	0.3	0.9	0.5	72.3	8.7	2.7	0.0	0.4	0.7	0.2	0.3	3.0	14.8	3.0	1.5
5	0.3	0.7	0.7	5.6	59.2	3.1	0.0	0.1	0.1	1.2	0.4	0.0	36.3	0.7	0.7
6	0.5	0.2	4.1	0.1	1.2	45.5	3.9	0.1	0.0	0.1	2.9	0.0	0.5	0.6	1.2
7	0.5	0.0	0.6	1.1	0.9	0.6	59.0	0.2	9.8	2.9	0.4	0.5	0.5	1.4	2.2
8	3.4	1.7	0.8	0.8	1.1	3.1	1.3	52.6	1.7	2.2	0.7	0.2	1.8	1.8	1.9
9	0.1	0.2	0.0	0.0	0.1	0.1	1.6	2.3	61.1	3.4	0.0	0.0	0.4	1.8	0.2
10	3.8	0.0	0.4	0.1	0.1	0.0	3.5	0.1	0.0	55.4	0.0	0.0	0.7	0.2	0.1
11	0.3	1.8	1.0	1.6	2.9	7.8	0.0	1.4	5.5	0.0	60.1	1.0	1.9	3.9	3.4
12	0.0	0.0	0.0	0.0	0.0	0.1	0.0	2.2	1.5	0.0	0.0	40.1	0.1	2.4	1.8
13	2.6	33.2	3.6	16.4	21.6	4.7	0.9	2.0	14.5	5.1	2.6	1.1	39.8	2.2	2.6
14	0.8	0.2	0.6	0.5	0.5	4.4	14.5	34.2	1.1	15.1	1.1	10.3	0.6	55.2	22.1
15	0.0	0.0	0.6	0.1	0.1	2.6	9.1	0.1	0.5	0.7	0.1	20.3	0.1	21.8	49.7
	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100

Table 19: Confusion matrix before uncertainty handling ($\tau = 0.0$) for all classifier modules trained on a collective data set. Overall recognition rate: 58.0%.

With the filtering upon the uncertainty value with a threshold of $\tau = 0.96$, the overall recognition rate can be boosted by 27.4 *pp* (*percentage points*) up to 85,4%. The corresponding confusion matrix is shown in table 20. This

	91%										75%				
	$\mathbf{M}_1$			$\mathbf{M}_2$			$\mathbf{M}_3$	$\mathbf{M}_4$			$\mathbf{M}_5$		$\mathbf{M}_6$		
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	99.7	0.0	0.4	0.1	1.5	2.2	0.4	0.0	0.0	1.1	0.4	5.1	0.0	1.5	5.2
2	0.0	90.9	0.1	0.0	0.0	2.1	0.0	0.0	0.0	0.3	0.3	0.0	0.4	0.7	0.7
3	0.0	0.0	98.9	0.0	0.0	5.1	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	3.0
4	0.0	0.1	0.0	92.6	1.5	0.6	0.0	0.0	0.0	0.3	0.0	0.2	6.1	2.2	0.7
5	0.0	0.1	0.0	1.0	49.5	0.1	0.0	0.0	0.0	0.0	0.0	0.0	17.0	0.7	0.0
6	0.0	0.0	0.1	0.0	0.0	88.1	0.0	0.0	0.0	0.0	0.0	0.0	0.4	0.0	6.7
7	0.0	0.0	0.0	0.1	1.5	0.0	98.2	0.0	1.3	3.4	0.0	0.0	0.0	2.2	7.5
8	0.2	0.2	0.0	0.0	0.5	0.3	0.0	98.3	0.1	0.0	0.0	0.0	0.4	2.9	3.7
9	0.0	0.0	0.0	0.1	0.0	0.1	0.2	0.6	98.3	0.3	0.0	0.0	0.0	2.2	0.0
10	0.0	0.0	0.0	0.0	0.5	0.0	0.5	0.0	0.0	91.3	0.0	0.0	0.0	0.0	0.0
11	0.0	0.0	0.0	0.0	1.0	0.1	0.0	0.0	0.1	0.0	99.1	0.0	0.4	0.7	0.7
12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	86.9	0.0	5.8	9.7
13	0.1	8.8	0.4	5.9	43.9	0.4	0.0	0.0	0.1	1.3	0.1	0.2	75.1	0.7	0.0
14	0.0	0.0	0.0	0.1	0.0	0.6	0.6	1.0	0.0	2.1	0.1	1.7	0.4	67.2	14.9
15	0.0	0.0	0.0	0.1	0.0	0.1	0.1	0.0	0.0	0.0	0.0	5.9	0.0	13.1	47.0
	69.3	60.3	73.7	15.1	6.2	26.5	56.8	42.1	50.3	27.7	40.8	25.0	9.2	4.4	4.6

Table 20: Theoretical optimum with knowledge of all training data. Filter threshold: $\tau = 0.96$. Overall recognition rate: 85,4%.

is a recognition rate for a system which can be used in real applications. But the filtering reduces the amount of classifications the recognition system has as output. This circumstance is discussed in detail in subsection 4.6.1. The percentage of remaining classifications for each class after filtering is displayed in each table in the last row of the confusion matrices. Since the classifier modules originally produce 12.5 classifications per second, a percentage of less than 8% remaining classifications could result in an effective output of less than one class per second. Usually, a person's activities do not change that fast, but it is still possible that this could result in missing an activity event. This effect needs to be analyzed in detail when the activity recognition system is used in a specific application.

Upon closer examination, one can see that most of the classes (9 out of 15) are recognized with an accuracy of over 90%. The classes with lower recognition percentages could have overlapping membership functions in the RFIS mapping, which is indicated through mutually misclassified classes. A good example for this circumstance is given by classes no. 5 and no. 13, where $43,9\%$ of data with the ground truth class no. 5 is misclassified on no. 13 and 17% from no. 13 on no. 5. These two provide a good example for classes whose patterns are hardly separable. The overall recognition rates in the evaluation of the extensibility are significantly lowered due to the fact that classes with overlapping membership functions are in the activity recognition system. It might be a good choice not to combine such classes in a classifier system. A solution to this problem could be a better sensor placement, which would lead to patterns which are easier to separate. This solution is not applicanble in a mobile phone based system, since the phone can only be carried on certain body postitons. A different approach might be to utilize data from different types of sensors for the separation, which is not part of this thesis.

Next, the classification accuracies when a new set of classifiers is added to a pre-existing one are examined. The original classifier set consisted of modules $\mathbf{M}_1$ to $\mathbf{M}_4$. To this, the set containing modules $\mathbf{M}_5$ and $\mathbf{M}_6$ is added. The results of the union without bit masking the classifiers are shown in table 21. Here, the recognition rate for the original modules is 95% as required. The added modules $\mathbf{M}_5$ and $\mathbf{M}_6$ are rarely activated, because the only case where they could be activated is when misclassifications occur and all original modules recognize the complementary class.

After the genetic algorithm has found a bit masking for the original classifiers, the recognition rates increase significantly (table 22): the overall rate for all classes is just 2 *pp* lower compared to the upper limit which is achieved when training all modules together. The still high recognition rates for the ten original classes indicate, that the original classification capabilities of these modules have not changed much. A recognition rate of 6 *pp* less compared to the non bit masked classifier modules and an increase to 15 recognized classes is a really good result compared to other approaches.

Summary and Discussion In the evaluation it has been shown that the modular combination of new and pre-existing classifiers have nearly the same recognition rates (only 2 *pp* less accurate) compared to a classification

	95%										17%				
	M_1			M_2			M_3	M_4			M_5		M_6		
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	99.3	0.2	1.7	0.4	0.8	2.8	0.5	0.2	0.2	4.5	8.6	0.0	0.8	7.7	2.3
2	0.0	98.9	0.9	0.4	0.1	0.6	0.0	0.1	0.2	0.7	73.8	1.2	0.3	16.9	6.8
3	0.0	0.0	92.9	0.0	0.0	0.2	0.0	0.1	0.0	0.0	0.0	26.0	0.0	26.2	34.1
4	0.0	0.0	0.0	86.4	2.1	0.0	0.1	0.0	0.1	0.0	0.0	0.0	4.1	1.5	0.0
5	0.0	0.3	1.9	12.9	95.4	0.2	0.2	0.1	0.3	1.5	0.0	0.6	92.2	20.0	18.2
6	0.0	0.0	1.7	0.0	0.6	95.4	0.0	0.0	0.0	0.0	0.8	0.0	0.2	6.2	6.8
7	0.2	0.0	0.0	0.0	0.0	0.0	99.0	0.0	0.0	11.2	0.4	0.0	0.0	0.0	0.0
8	0.3	0.2	0.6	0.0	0.4	0.7	0.1	99.4	0.0	2.2	0.0	15.0	0.9	13.8	20.5
9	0.0	0.0	0.2	0.0	0.0	0.0	0.2	0.1	99.2	0.0	0.0	1.2	0.3	0.0	2.3
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	79.9	0.0	0.0	0.2	0.0	0.0
11	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	16.4	0.0	0.0	3.1	0.0
12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	55.5	0.0	1.5	2.3
13	0.0	0.4	0.0	0.0	0.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.2	0.0	0.0
14	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.6	0.0	3.1	0.0
15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	6.8
	51.0	75.7	14.3	29.4	26.9	21.4	59.1	79.3	67.5	9.8	7.5	3.8	22.0	2.1	1.5

Table 21: Modular approach without the need for knowledge of all training data in all training steps that is not masked with a bit vector. Threshold: $\tau = 0.96$. Overall recognition rate: 68,6%.

	89%										73%				
	M'_1			M'_2			M'_3	M'_4			M_5		M_6		
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	99.6	0.1	0.3	0.2	1.4	3.1	0.3	0.0	0.0	0.8	0.3	5.1	0.0	1.5	5.8
2	0.0	92.1	0.1	0.0	0.0	2.7	0.0	0.0	0.0	0.0	0.2	0.1	0.4	0.8	2.2
3	0.0	0.0	96.1	0.0	0.0	5.9	0.0	0.0	0.2	0.0	0.0	0.0	0.4	0.0	2.9
4	0.0	0.2	0.2	87.2	7.7	0.3	0.0	0.1	0.0	0.3	0.1	0.3	12.4	3.8	1.4
5	0.0	0.1	0.1	1.6	46.4	0.2	0.0	0.0	0.0	0.0	0.0	0.0	16.3	0.0	0.7
6	0.2	0.0	2.7	0.0	0.0	84.9	0.0	0.0	0.0	0.0	0.0	0.0	0.8	0.0	5.8
7	0.0	0.1	0.0	0.2	1.9	0.0	98.1	0.0	2.1	1.4	0.0	0.0	0.0	2.3	7.2
8	0.1	0.2	0.0	0.0	0.5	0.3	0.0	98.4	0.1	0.0	0.0	0.0	0.4	3.1	3.6
9	0.0	0.0	0.0	0.2	0.0	0.2	0.2	0.6	97.3	0.5	0.0	0.0	0.0	2.3	0.0
10	0.0	0.0	0.0	0.0	0.0	0.0	0.6	0.0	0.0	94.8	0.0	0.0	0.0	0.0	0.0
11	0.0	0.0	0.0	0.0	1.0	0.2	0.0	0.0	0.1	0.0	99.1	0.0	0.0	0.8	0.7
12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	86.4	0.0	6.9	8.7
13	0.1	7.4	0.5	10.0	41.1	0.6	0.0	0.0	0.1	0.8	0.2	0.2	69.0	0.0	0.0
14	0.0	0.0	0.0	0.2	0.0	1.0	0.8	0.9	0.0	1.4	0.1	1.5	0.4	64.1	13.8
15	0.0	0.0	0.0	0.2	0.0	0.6	0.1	0.0	0.0	0.0	0.0	6.5	0.0	14.5	47.1
	69.0	59.9	65.7	7.9	6.6	24.6	57.9	43.2	47.5	26.8	42.4	24.6	8.6	4.2	4.8

Table 22: Modular combination without the need for knowledge of all training data in all training steps, masked with a bit vector. Threshold: $\tau = 0.96$. Overall recog. rate: 84,0% (optimal 85%).

system, where all modules are trained together. To enable the transition of old modules onto new ones, a bit-vector for masking the old modules has to be provided.

The identification of these bit-vectors also demands some calculation time, but compared to a completely new training of all the old and new classifier modules the computational time is negligible. Since both the training of the classifier modules and the identification of the bit-vectors include an indeterministic heuristic search, no numbers about the runtime or the complexity can be given. As the training includes two complex clustering algorithms, a genetic algorithm search, linear regression and two loops (inner and outer), the runtime of the training is expected to be much more calculation and time intensive than the bit-vector identification. For the bit-vector identification for each generation the fitness upon the training data needs to be calculated, which is just the determination of the classification per module. This can be optimized using multi core or highly parallel graphics processors, which are not very probable with the clustering algorithms used in the training.

Also, the bit-vector masking preserves the capabilities of the old classifier modules, since their recognition rates only drop 6 *pp* compared to non masked modules, and it is easily removable by using a bit-vector for masking which is all ones. The recognition rates of masked old modules and unmasked modules of the referenced upper limit, where all modules are trained together, are just 2 *pp* apart. This results strongly argue for the extension of

queues than for retraining the whole activity recognition system.

Additionally to the evaluation results of the extendibility other facts revealed itself. The filtering on the uncertainty value achieves a boost of recognition up to 95%, which in the end makes activity recognition practical. This particular circumstance is evaluated, analyzed and discussed in detail in subsection 4.3.

5.3 Evaluation: Challenge 3 - Robustness

As **robust** activity recognition a classifier has been defined that is capable of classifying the noisy and varying accelerometer sensor patterns of mobile phones. For that purpose the mapping function of the classifier modules is recurrent, so the output of the mapping is fed back as additional input. This kind of mapping function not only stabilizes the classification process, but also enables the derivation of a reliability measure. The reliability measure can be used in a filtering process to improve the overall statistical accuracy and therefore provide further **robustness** to the activity recognition on mobile phones.

First, the filtering upon a reliability measure and the robustness through the RFIS mapping function is evaluated in this subsection. The feedback of the mapping outcome as additional input value can be evaluated without any other improvement to the classifier, which are described in this thesis. The RFIS mapping function is compared with a non recurrent FIS mapping. Results will show, that a recurrent FIS mapping is superior to a non recurrent one.

Last, the filtering upon the reliability measure is evaluated on how it can improve the statistical accuracy of the modular activity recognition and therefore the robustness of the classification process. The evaluation of the filtering is done upon the proposed modular classifier structure, which includes all improvements presented so far.

5.3.1 Fuzzy Classifiers vs. Recurrent Fuzzy Classifiers

The evaluation in this subsection first considers the improvements in accuracy and robustness, which were made through taking the mapping function's output and feed it back as additional input. The evaluation results will show, that even systems detecting many activities can have a high accuracy with a recurrent classifier.

Evaluation Setting To compare the accuracy of recurrent and non recurrent mapping functions in a classification process, accelerometer data from the OpenMoko Freerunner phone is used. As discussed before, the recognition with a RFIS mapping function should stabilize the classification and improve the overall accuracy. The evaluation setting includes the recognition of five activity classes with the usual preprocessing of 8-sampled windows with mean and variance. The classes used are listed in the following table:

"lying still"	(class no. 1)	$\Rightarrow$	no movement of device
"knocking appreciation"	(class no. 2)	$\Rightarrow$	knocking on table with device next to it
"sitting"	(class no. 3)	$\Rightarrow$	device in users pocket whilst sitting
"standing"	(class no. 4)	$\Rightarrow$	device in users pocket whilst standing
"walking"	(class no. 5)	$\Rightarrow$	device in users pocket whilst walking

Data was recorded on several controlled test runs with five subjects. A sequence of the classes was simulated to reflect a conference event. These results have been previously published in [23].

The architecture of the activity classifier differs in two aspects compared to the one presented in subsection 3.1 and usually used in this thesis. The fuzzy mapping function used in this evaluation implements non covariant membership functions as described in equation 7. This is due to the original setting of this evaluation, where not only accelerometer data was classified, but also audio data, which is not as strongly correlated. Also, instead of many modules only one monolithic classifier was used, since the effects of the RFIS mapping function should be evaluated independently.

The feedback of the RFIS stabilizes the classification process significantly. The most erroneous classifications are made when there is a change from one class to another one. To evaluate this disadvantage a check data set is used that reflects this insufficiency. The check data set consists of subsets ($\sim$30 data pairs each) of class specific patterns (many subsets per class), which where randomly ordered: OpenMoko Freerunner phone with two 3-axis accelerometers - 1410 training data pairs (352,5 sec) $\rightarrow \sim$47 successive class changes; 1770 check data pairs (442,5 sec) $\rightarrow \sim$59 successive class changes. Therefore, each activity has a period length of 2.4 seconds which, compared to normal human behavior, is a stress test for the RFIS mapping function.

Evaluation Results The feedback before the first classification is zero, which is not identifying any class and therefore does not benefit the recognition of any particular activity. Despite these challenges the RFIS performs significantly better for all three classifiers than the standard FIS. The confusion matrices for both classifiers, FIS and RFIS classifiers, are shown in table 23. Table 23 shows the results of the accelerometer data classifier, where the overall correct classifications of the FIS are ∼62% and for the RFIS ∼94%. This shows an improvement of about 32*pp* (*percentage points*). The most significant improvement is observed for the class no. 1, which denotes

designated classes	classes classified onto 1	2	3	4	5
1	0	100.00	0	0	0
2	0.74	99.26	0	0	0
3	0	8.33	90.00	1.67	0
4	0	0	0.3344	99.67	0
5	2.33	0.33	0.33	10.33	86.67

designated classes	classes classified onto 1	2	3	4	5
1	90.50	3.00	2.33	4.17	0
2	0.74	96.67	2.5926	0	0
3	0	0.33	94.67	3.67	1.33
4	0	0	0	99.67	0.33
5	0	0.33	0.67	8.00	91.00

Table 23: Confusion matrix of FIS ∼62% (left) and RFIS ∼94% (right) mapping function in activity classifier.

the activity *lying still* for the phone on a table. This class strongly interferes with class two *knocking appreciation* where the user knocks on the table next to the phone. One hundred percent mis-classifications is surprising though. This might be an indication, that the sensor in the used phone might not be sensitive enough and the classification results for this classes could also be vice versa. An improvement of up to over 90% strongly argues for the recurrent mapping function. Here the feedback of the mapping outcome totally stabilizes the classification for class one, but lowers the classification accuracy for class no. 2. Since the degredation is with about 4*pp* not significant, the RFIS can be seen as superior over the FIS mapping.

The accuracy of the rest of the classes no. 3 to no. 5 is improved. The class no. 3 by over 4*pp*, no. 5 also by over 4*pp* and the false positive classifications of class no. 4 shifted from class no. 3 to no. 5. Again, the results indicate a predominance of the recurrent classification over the non recurrent one. This conclusion is also supported by the results for a recurrent audio classifier. The results of this recurrent audio classifier can be found in [23].

5.3.2 Filtered vs. Non-Filtered RFIS Classification

Now the filtering upon the reliability measure is analyzed how it improves the statistical accuracy. The filtering upon the reliability measure can not be analyzed using a system without recurrence, since the desired reliability measure can only be derived with a recurrent classifier. Since the feedback of the output of the mapping function as additional input also stabilizes and improves the recognition rates of the classifier modules, this effect cannot be eliminated in this evaluation.

Evaluation Setting For further evaluation again the accelerometer data of the OpenMoko Freerunner phone is used. The data is provided by two different test subjects. The data is processed, used for training and classified offline in Matlab. Three different classifier modules are trained on nine different activities in total, where each module is used to classify on a subset of three classes. The combinations of activity classes per classifier and *conditional contexts* are listed in table 24. For training of each class a data set of 1500 annotated feature vector pairs and 3000 for each of the complementary class is used. The data of the complementary classes consists of training data of the respective other modules, as explained before. The check data for evaluation of the training consists of 500 data pairs for each of the native classes and 1000 per complementary class. The rest of the data is used for evaluation of the filtering approach. The evaluation data has about 2000 data pairs per class, which should provide valid evaluation results.

Evaluation Results Not the accuracy of each classification can be improved due to the filtering, but statistically over time the amount of correct classifications can be improved. This is due to the separation of the reliable from the unreliable classifications according to the reliability measure. Therefore many incorrect classifications can be filtered out as well as some correct classifications. This reduces the total amount of classifications, where the

Conditional Context	Context Class	Class No.	Classifier Module
Phone in users trouser pocket: *no movement*	user is sitting user is standing user is lying	1 2 3	$\mathbf{M}_1$
Phone in users trouser pocket: *movement*	user is walking user is climbing stairs user is cycling	4 5 6	$\mathbf{M}_2$
Phone in users hand:	just holding talking on phone typing text message	7 8 9	$\mathbf{M}_3$

Table 24: Activity classes, *conditional contexts* and classifier modules for the acceleration sensor.

distribution of the passed classifications is unknown. There might be periods where nearly all classifications pass the filter and others in which none get through at all. The tradeoff between statistical accuracy and activity event detection rate - definition follows in the respective subsection - is analyzed in subsection 4.6.1. In the following confusion matrices the percentage of remaining classifications is always listed in the last row.

To have a good tradeoff between false and true negative classifications, the Receiver Operator Characteristic (ROC) can be used to determine the threshold value. Details on this approach can be found in [54] or in subsection 3.1.5. In this evaluation a different approach is used, where the overall classification accuracy is plotted according to the filter threshold in figure 39. Since the most incorrect classifications of a recurrent classifier are made when

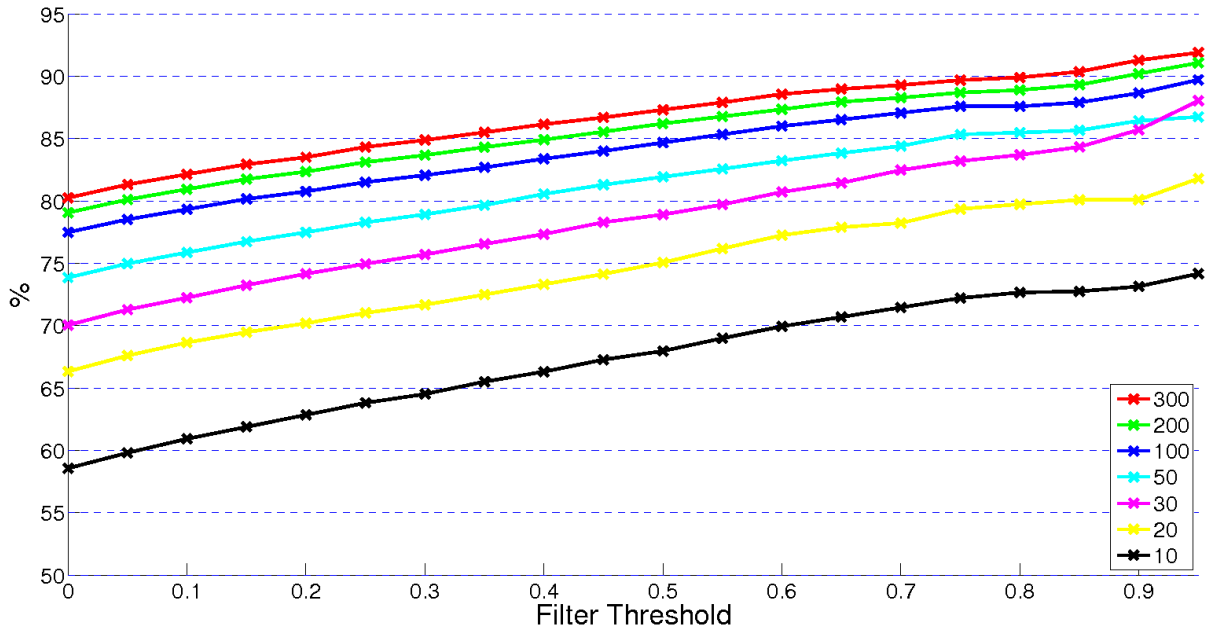


Figure 39: Graphs of percentage of overall correct classifications according to the filter threshold for different activity period lengths.

the actual activity changes, the overall classifier accuracy strongly depends on the period length of the activities. Different activities in real life have different period lengths, e.g. activities like *walking* or *standing* are shorter than *sitting* or *lying*. This certainly depends on the profession or the hobbies of the user. If she is e.g. a bakery sales women or likes to hike, then the previous assumption is no longer valid. Due to this circumstance different period

lengths are used for evaluation. The evaluation data of one period length is randomized in slices of the respective length, so the activities sequence is not periodic to a certain pattern, which would again influence the results. For evaluation the period lengths 10, 20, 30, 50, 100, 200 and 300 of successive classifications for one activity were used. With a sampling rate of 100Hz and a window size of 8 samples 12.5 classifications are made in average per second, the periods correspond to lengths of 0.8, 1.6, 2.4, 4.0, 8.0, 16 and 24 seconds. Each of the different activity period lengths are plotted as separate graph in the figure 39.

The results show the expected, that the longer the period length and the higher the filter threshold, the better the overall classification accuracy is. A period length of 10 successive feature vectors for one activity is a stress test for the recurrent classifier architecture. In this case the feedback has the lowest stabilizing effect on the classification, since when the classifier has engaged on the new activity already feature vectors for a different one need to be classified. The graphs for the next higher period lengths are significantly above the graph for a period length of 10, where the graphs of the very high period lengths of 100, 200 and 300 are very close to another. With the higher period lengths the improvement through the RFIS mapping reaches its upper limit.

The graphs in figure 39 also show, that the filtering significantly improves the overall accuracy of the modular activity recognition. An example of a lower period length of 30 successive classifications is selected for detailed analyzes. Here the overall recognition accuracy improves from no filtering to a filter threshold $\tau = 0.95$ by 18*pp* (*percentage points*). The system with no additional filtering has with 70.0% correct classifictions nearly no chance to be applied in a real world application, whereas the same classifier with a fitering on the threshold $\tau = 0.95$ classifies 88.0% of the feature vectors correctly or filters them out.

Detailed Evaluation Results A closer look into the evaluation results of the period length 30 offers the confusion matrices listed in table 25. Here one can see, that the classification rates for the unfiltered classes are mostly below, where only two are classified with above 80% accuracy. There is also one class, activity *user is climbing stairs* with class no. 7, where only an accuracy of 50% could have reached. This class strongly intgerferes with the activity *user is sitting* with class no. 1. A masking of the modules $\mathbf{M}_1$ and $\mathbf{M}_2$ with bit-vectors could improve the accuracy of this classification. The process of a bit-vector masking was described in subsection 3.1.6 and analyzed in subsection 4.1. Nevertheless, the accuracy of classifications are also improved for class no. 7 through

	1	2	3	4	5	6	7	8	9
1	77.2	13.5	2.0	6.2	6.6	7.2	30.6	3.0	7.2
2	4.1	56.3	3.7	1.3	0.6	0.8	0.5	0.1	7.9
3	0.1	0.2	85.9	0.0	0.0	0.1	0.0	0.0	0.0
4	2.2	10.8	0.8	65.2	25.6	6.2	6.6	1.9	2.7
5	1.7	5.4	1.1	23.8	62.5	4.1	2.4	0.3	0.2
6	0.1	4.4	2.2	0.3	0.5	76.7	0.8	0.1	0.0
7	12.0	4.0	4.0	2.9	3.1	4.0	49.9	6.1	3.4
8	1.5	5.1	0.3	0.4	0.9	0.4	4.4	87.9	10.7
9	1.2	0.2	0.1	0.0	0.1	0.1	0.9	0.6	67.9
	100	100	100	100	100	100	100	100	100

	1	2	3	4	5	6	7	8	9
1	92.5	3.6	0.1	10.7	7.8	3.1	22.0	0.2	2.2
2	1.1	90.7	0.0	1.2	0.0	0.0	0.0	0.0	2.2
3	0.0	0.0	99.6	0.0	0.0	0.0	0.0	0.0	0.0
4	0.4	2.5	0.0	78.6	3.3	0.8	2.0	0.0	1.3
5	0.4	0.5	0.1	6.0	83.3	2.7	1.3	0.0	0.0
6	0.0	0.0	0.1	0.0	0.0	90.3	0.0	0.0	0.0
7	5.0	0.3	0.1	3.6	5.6	3.1	68.7	0.4	1.3
8	0.0	2.5	0.0	0.0	0.0	0.0	0.7	99.4	3.5
9	0.7	0.0	0.0	0.0	0.0	0.0	5.3	0.0	89.5
	15.0	15.2	72.0	3.0	3.1	16.9	8.0	77.6	10.3

Table 25: Confusion matrices for non filtered (left) with 70.0% and filtered activity recognition (right) on threshold $\tau = 0.95$ with 88.0% overall accuracy; period length of each activity are 30 successive classifications.

the filtering, which is shown in the confusion matrix in table 25 on the right side for a threshold $\tau = 0.95$. The amount of correct classifications for this class has improved by 18.7*pp* to 68.7%. This is still not an accuracy for an applicable classification, but shows the signifciant improvement through the filtering. The class no. 7 is the exception anyway, where the majority of the activity classifications could be improved to over 90%.

As explained before, the filtering reduces the amount of classifications passed to a next instance of processing or application. The percentages of remaining classes are listed in the last row of the confusion matrices for each class individually. For some of the classes only a small percentage of the classifications remain, such as for the classes no. 4 and 5, where only 3% are passed to the next level. The tradoff between filtered out and passed classifications can be influenced through the threshold. The user has to decide if the accuracy or the classification frequency are more important for her application. The tradoff between this two goals is analyzed in subsection 4.6.1.

The second closer look is done to the evaluation results of the modular activity recognition for a sequence size of 100 successive classifications. The confusion matrices for no filtering (left) and filtering on a threshold $\tau = 0.95$ (right) are listed in table 26. The improvement in this case is not as much as with a period length of 30, but still a

	1	2	3	4	5	6	7	8	9
1	83.4	12.2	2.9	3.8	2.1	1.9	25.4	1.5	3.3
2	5.7	69.1	3.2	1.4	0.2	0.6	0.5	0.1	4.4
3	0.1	0.3	88.9	0.0	0.1	0.0	0.0	0.0	0.0
4	1.7	6.7	0.5	67.1	24.6	4.3	2.8	0.1	1.1
5	0.5	3.0	0.8	25.5	70.8	3.9	0.8	0.1	0.3
6	0.0	0.9	2.5	0.4	0.5	87.1	1.0	0.0	0.0
7	7.3	4.4	1.2	1.5	1.3	2.0	59.9	6.7	1.9
8	1.2	2.8	0.0	0.2	0.4	0.2	4.8	90.9	9.5
9	0.2	0.5	0.0	0.0	0.0	0.0	0.9	0.6	79.5
	100	100	100	100	100	100	100	100	100

	1	2	3	4	5	6	7	8	9
1	92.4	2.6	0.1	2.6	3.3	0.7	22.3	0.1	1.1
2	1.3	91.5	0.1	5.2	1.1	0.4	0.0	0.0	2.2
3	0.0	0.0	99.6	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	1.7	0.0	76.6	2.2	0.7	0.6	0.0	0.7
5	0.0	0.4	0.0	10.4	87.9	1.5	0.0	0.0	0.0
6	0.0	0.0	0.1	1.3	0.0	95.6	0.0	0.0	0.0
7	6.3	0.7	0.0	3.9	4.4	1.1	72.0	0.6	0.4
8	0.0	2.6	0.0	0.0	1.1	0.0	0.0	99.3	3.0
9	0.0	0.4	0.0	0.0	0.0	0.0	5.1	0.0	92.6
	16.2	19.0	76.6	2.8	3.1	18.0	8.4	81.6	12.2

Table 26: Confusion matrices for non filtered (left) with 77.4% and filtered activity recognition (right) on threshold $\tau = 0.95$ with 89.7% overall accuracy; period length of each activity are 100 successive classifications.

plus of 12.3*pp* is gained. Already the unfiltered classifications have an overall accuracy of 77.4%, which is 7.4*pp* higher as for the smaller sequence size. After the filtering the accuracy is 89.7%, which is close to the accuracy of the activity classification for the 30 samples length of 88%. The lowest accuracy of 72% is again noticed for the class no. 7. The evaluation data of this class is just too similar to the data of class no. 1. In this case the class no. 1 is favoured over the class no. 7, but this could be vice versa. The position of the mobile phone in the users pocket could be adjusted to cope with this problem, but this would need an adaption of the user behaviour, which is not possible in real life.

The evaluation results show, that filtering can improve the statistical accuracy of the modular activity recognition, which also reducing the amount of passed classifications. This tradeoff is analyzed in detail in subsection 4.6.1. In the end the user has to decide which is the most important. Also, an additional fuzzy reasoning process to the modular activity recognition could use the reliability measure to further improve accuracy while aggregating higher conclusions, like analyzed e.g. in [23]. This would make the filtering obsolete.

5.4 Evaluation: Challenge 4 - Resources

Since the **resources** on mobile phones are limited, this challenge is a very important one. The challenge divides the **resources** into the two relevant ones: processor and battery. In case of the processor load due to the activity recognition, the late developments in mobile CPUs ease this challenge. The battery runtime is another issue, since the capacity and size can not be improved much further. Due to the proposed modular activity recognition the processor load could be limited to a minimum without the loss of accuracy. Reducing the calculation effort also decreases the power consumption. To further reduce the power drain due to the activity recognition a sleep-time scheduling has been introduced.

The evaluation of the computational effort, the processor load and the energy consumption is subdivided into four parts according to the different evaluation aspects.

First, initial evaluation results are presented. For this evaluation the accuracy of a monolithic classifier is compared with subdivided modular one, which is built according to a divide and conquer approach. Furthermore initial experiments with classifiers scheduled in a dynamic queue are presented. The experiments are still based on data from the AwarePen artifact, but easily transferable to mobile phones. Also, the influence on the accuracy of the classification due to the recurrent fuzzy mapping function and the covariant membership functions are eliminated, to have a clear conclusion on the modularity effects.

Second, the performance of the classifier modules in a dynamic queue is evaluated in detail. Here the architecture as proposed before is used. The accuracy and calculation effort for different subdivision of the overall activity classes into different modules is compared. Also, the effort for calculating the different subdivisions is derived according to the mean evaluated rules and the input dimensionality.

Last, the energy consumption of the activity recognition on the OpenMoko Freerunner phone is measured. Also, the energy consumption of the activity recognition with sleep-time scheduling is measured. Here the accuracy is compared towards a so called activity event detection rate.

5.4.1 Initial Experiments

Initial experiments in this evaluation of the solutions to the challenge resources the advantages of a modular classifier structure are analyzed. In particular, the complexity of the mapping function is analyzed in this subsection. The basis of the initial experiments is a monolithic classifier structure, which is successively divided into subparts. In the end of this analysis the used dynamic queue of modular classifiers results.

The mapping function used in this initial experiments is not the one described so far, as it uses multivariate (eqn. 7) instead of covariate membership functions (eqn. 10), the output is not fed back as additional input and the subtractive clustering (subsec. 3.3.2) clusters on the whole set of training data not the class specific subparts separately. This is to eliminate the effects of these improvements so that the modularity can be analyzed independently. The FIS training algorithm also differs from the one described in subsection 3.3, since the missing recurrence and covariant membership functions results in a simpler algorithm. Here the original training algorithm proposed by Jang [81] is applied.

The experiments and analysis presented in this subsections had previously been published in [29]. The source of sensor data is sill the AwarePen artifact [27], but the results of the analysis are transferable to mobile phones, since the accelerometer sensor is equal in behavior for both platforms.

Monolithic vs. Modular In [27] was shown that a FIS is capable of detecting even complex context information from sensor readings. Nevertheless, handling and detection of classifications can be improved by dividing the processing of a FIS into several subparts. Throughout this initial experiments such a FIS is step-by-step *divided* into subparts to maximize classification results and to keep the calculation effort to a minimum. The division is not done literally, instead of dividing the amount of classes each FIS should map onto. At first, a single FIS with a varying number of rules, represents all classes. The activity classes are: *(1) lying on table, (2) cell ringing next to artifact, (3) writing horizontally, (4) writing vertically, (5) playing around with the pen, (6) pointing at sth., (7) having the pen in the pants pocket while sitting, (8) standing and (9) walking*. In the next step, two FIS do the same mapping, also with a variable set of rules each. In the last part of this initial evaluation the mapping is done via four dynamically queued FISs.

The usage of one FIS - without recurrence and covariant membership functions - for mapping onto nine context classes with a reasonable amount of rules is nearly impossible. Separating the patterns from acceleration sensors is difficult when the patterns are too similar. For example, separating the patterns of *writing horizontally* and *pointing at sth.* is hardly possible with only mean and variance data, because they are nearly the same for both. In fuzzy set theory it has been proven [146] that every function can be represented to infinite precision with a FIS with an infinite number of rules, but such theory is especially of low value for resource limited embedded or mobile systems. The chosen clustering shows best results with a low amount of fuzzy rules, but still is not able to separate the patterns to a satisfying degree. The percentage of correct classified data pairs for a varying number of rules is shown in figure 40 as gray curve and $\times$ markers.

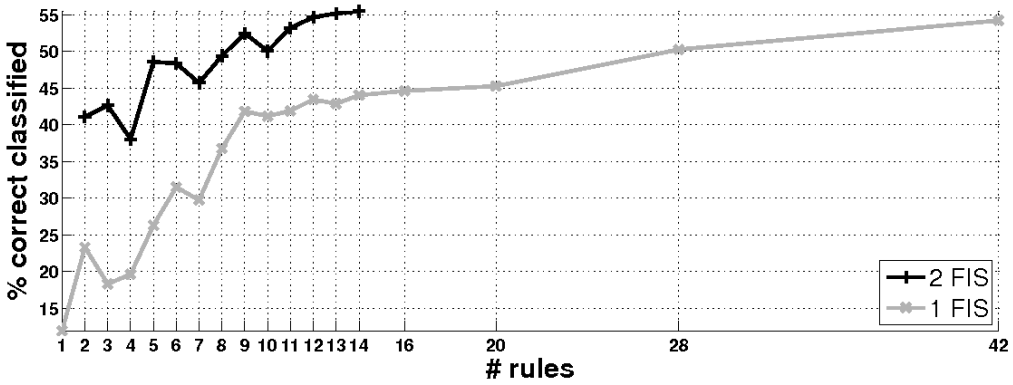


Figure 40: Percent of correct classified data pairs for one FIS classifying all classes (gray $\times$) or two FIS mapping onto classes (black $+$)

A better result than a monolithic approach can be achieved when the big FIS representing all classes is divided into several FISs. The initial experiments start here by explaining how to separate the monolithic FIS into two modules, and later it is explained how to apply this method to an n-FIS system.

In the first example, each FIS maps onto four (FIS A) or five (FIS B) basic activity classes (as *writing*, *playing* etc., see fig. 41). To allow the transition from one FIS to another, an additional class is introduced that represents all classes, which are classified by the other module. This special class is called *complementary class*. To train each FIS correctly, an equal amount of data pairs for basic classes and for the *complementary class* is required. In the two-FIS example, the training data for the *complementary class* consists of data for all classes that the second FIS is mapping on. Correct selection of the training data is important, as performance depends on the correct detection of basic classes, but also on correct detection of the *complementary class*. The recognition percentage for an average rule evaluation based on a test data set is plotted in Fig. 40 (black curve with $+$ markers) against the equal amount of rules for the monolithic FIS classifier recognizing all context classes. For this plot the order giving the best classification result is chosen for the example (first FIS A than FIS B).

For evaluation purposes a different amount of rules for each FIS are combined and analyzed upon the average recognition rate for a test data set. The recognition rate in percentage is plotted as a surface in figure 42 on the left and as a contour plot on the right. A brief analysis of these plots can be found in the discussion of this subsection.

To implement each of the two FISs above, it is important to know the number of rules that have to be processed by each FIS for correct classification. Due to the high variance of possible input data, it is difficult to give a general estimation of the mean number of rules. A mean estimation for the number of rules to be processed for an input data can be given:

$$\begin{aligned} N_{AB} &= \mathbb{P}(A)(n_A + \mathbb{P}(B|A)n_B) + \mathbb{P}(B)(n_B + \mathbb{P}(A|B)n_A) \\ &= n(\mathbb{P}(A)\mathbb{P}(B|A) + \mathbb{P}(B)\mathbb{P}(A|B)), \text{ for } n_A = n_B = n \text{ and } \mathbb{P}(A) + \mathbb{P}(B) = 1 \end{aligned}$$

Probabilities are only distinguished by the amount of context classes represented by each FIS and not through the average recognition rate for the classes or the complementary one. Hereby $\mathbb{P}(x)$ with $x \in \{A, B\}$ is the probability

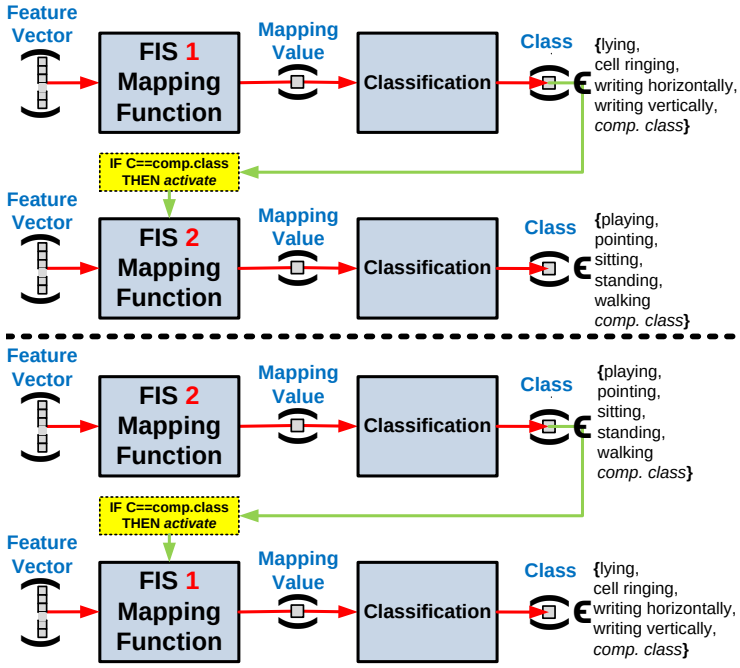


Figure 41: Schematic of activity recognition with two statically queued FISs - (top) combination with 50% and (bottom) with 48% correct classifications

of the FIS x to be evaluated according to the number of classes it classifies on. The probability that the second FIS $Y \in \{A, B\}$ needs to be executed - if the first x is not capable of classifying the data - is expressed through the formula $\mathbb{P}(Y|x)$. The number of rules each FIS consists of is n_x with $x \in \{A, B\}$, where in this case $n_A = n_B$. If each FIS is handling the same amount of rules, the mean rule evaluation is $\frac{1}{2}n$.

Four Modules Classifier The above approach can be continued by further subdividing the classification into four FISs. For example, one FIS (FIS_{lying}) represents *lying on table* and *cell ringing next to pen*, the next one (FIS_{hand}) classifies *playing* and *pointing*, another one (FIS_{write}) maps *writing horizontally* and *writing vertically* and the last (FIS_{pants}) manages the classes *sitting*, *standing* and *walking*. The FISs are statically queued after another. The average amount of rule evaluations of 16,15 rules is a bit high, but the recognition rate has improved up to 68,44%.

Up to this point the best order of the FISs in a queue was based on the overall classification accuracy of the different sub-classifiers from a test-data run, and therefore static at run-time. Even better results can be reached using a dynamic ordering of FISs based on current state of the stochastic process. Basic knowledge about the statistical behavior of the classifiers and the underlying process that is classified, allows the developer to optimize the order of the FIS queue dynamically, and avoid unnecessary execution of FIS modules. Although by adding some complexity here, it can be shown, that not only resources can be saved, but also the recognition rates can be improved.

One discovery which was made looking at typical AwarePen states in a typical usage process is, that the transition of a classified state to itself during following classification is more likely than to any other state. Furthermore, one can also see that some classes, like *playing* after *writing*, have higher transition probabilities than for example *walking* after *writing*. In order to make use of this statistical feature in these initial experiments, the classes are intuitively grouped in a way that the similarity form a kind of *macro state*. The transition probability from this state towards a state with classes from another FIS is much lower than towards itself. Also, this *macro state* can include meta knowledge, which e.g. can indicate the position of the device or artifact. Therefore the *macro state* is called *conditional context*.

This property is used to pre-compute an order of a classification queue, such that the classifier is always executed first that matched in the last step. This way the expected number of rule executions is reduced and the recognition

		classified onto class									
		err	1	2	3	4	5	6	7	8	9
designated classes	1	0	94.1	5.0	1.0	0	0	0	0	0	0
	2	0	30.9	69.1	0	0	0	0	0	0	0
	3	5.8	3.8	1.9	63.5	3.8	13.5	7.7	0	0	0
	4	0	1.0	0	12.5	80.2	6.3	0	0	0	0
	5	2.1	1.1	1.1	6.3	0	87.4	2.1	0	0	0
	6	1.0	1.0	1.0	1.0	2.9	15.2	78.1	0	0	0
	7	0	0	0	0	0	0	0	100.0	0	0
	8	0	0	0	0	0	0	0	0	100.0	0
	9	0	0	0	0	0	0	0	0	4.3	95.7

Table 27: 4 FISs dynamically queued - Confusion matrix

rates are improved since it becomes less likely to make mistakes in previous classifiers. The results of such a scheme are shown in the confusion matrix in table 27. The recognition rate is again improved and the dynamic FIS queue classifies 85% of the check data correctly.

Summary and Discussion As one can see from these initial experiments, interesting trade-offs between execution time and classification accuracy can be found. Understanding those trade-offs in relation to the underlying statistical process contains a high potential for a model-guided optimization. Analyzing the cost function of subsection 4.4.1 one can see different possibilities for minimization. In the simplest case above, that splits a FIS into two separate classifiers, where only the execution order of the classifier modules was changed. Because only static queues were used - i.e. the classification process is independent on $\hat{x}_t$, and $\mathbb{P}(x_t)$ was equally distributed - the expectation for the cost is solely dependent on the recognition rates of the correct and *complementary class*, as already seen in figure 41.

The second degree of freedom is modifying the internal FISs classifiers themselves by changing the rule numbers and thus also adapting the execution cost expectation. The resulting optimization space is plotted in figure 42 comparing recognition rates depending on the number of rules in a classifier that show a certain degree of linearity in the distribution, which suggests that classifier systems can be modeled statistically. One can also see that the

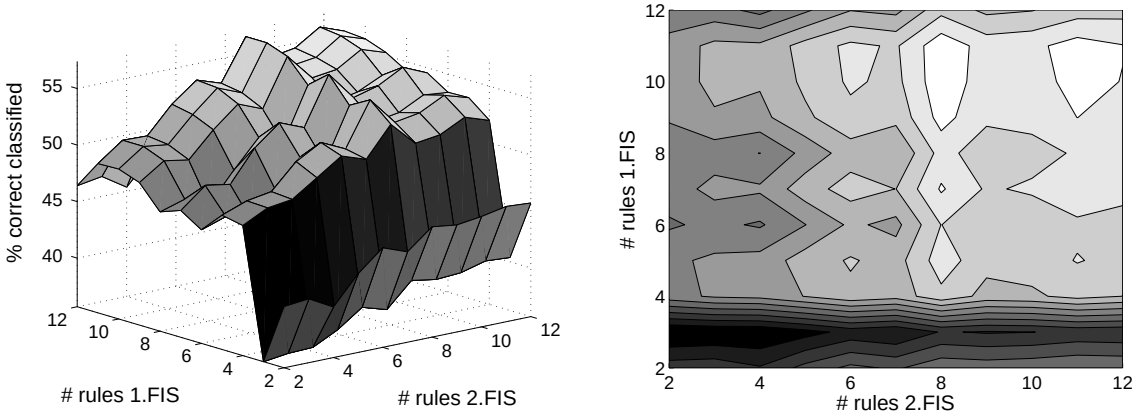


Figure 42: Surface plot (left) and contour plot (right) for percent of right classified data pairs for two FIS - better FIS first in line

maximal recognition rate is roughly distributed around the ”natural” cluster amount returned by the initial subtrac-

tive clustering. Knowing this optimal number of rules and the according recognition rate allows the estimation of the recognition rates of FISs with a different number of rules. This would mean, that even when the optimization space already grows quickly for the static case, good solutions can easily be found. The figure also shows that Pareto optimal solutions can be found for cost (proportional to the distance to zero) and recognition rate (height/color).

The third degree of freedom which is discussed here and represented in the model is the grouping of classes by common or separate classifiers. Naturally this affects the local recognition rates, as seen already in 40. In the dynamic queue approach, which can change the queue order by a pre-computed lookup based on the previous match, the grouping additionally affects the joint probability for having to detect previous *complementary classes*. This probability would be 1 if $\mathbf{M}_{m(x_t)} = \mathbf{M}_{m(\hat{x}_t)}$ and thus optimal, if the previously classified state has the same classifier as the current. The potential to heuristically optimize such dynamic cases based on the model can be seen. By grouping classes into classifiers by their transition probabilities at the beginning of the last section, the observation that $\mathbb{P}(x_t|\hat{x}_{t-1})$ is dominated by $\mathbb{P}(x_t|x_{t-1})$ for high overall recognition rates and this fact was then exploited. A model-guided optimization can also provide the highest flexibility, since the developer does not have to worry about how the activity classes are grouped in the modules. The results shown in 27 already hint at the potential for such an approach.

The results of these initial experiments show that the modular approach offers not only an activity recognition solution which needs less computational resources, but also improves the accuracy of the classification process. With the modularity not only the challenge computational **resources** are coped with, but also the challenges of **flexibility**, **extensibility**, **conditionality** and indirectly energy **resources**. Therefore, these initial experiments, which were conducted early in the practical research for this thesis, had already indicated the huge success of the modular approach in activity recognition on smart artifacts and mobile phones.

5.4.2 Complete Analysis of Dynamic Queue Complexity

The initial experiments of the modularity with respect to complexity and accuracy were conducted on the basis of non recurrent mapping functions with multivariate membership functions. Since the architecture proposed in this thesis includes these features in conjunction with a novel RFIS training algorithm, the modularity is now analyzed with the proposed modular activity recognition.

Evaluation Setting An evaluation is now set up, which uses accelerometer data from the OpenMoko Freerunner phone of one user. A varying amount of modules was used to classify onto twelve activity classes, since twelve classes are dividable into two, three, four, six and twelve modules. The activity classes for the varying amount of modules are listed in table 28. The basis of the evaluation is again the monolithic classifier, where all activities are

Context Class	STR	No.	Mod.
sitting	PSI	1	$\mathbf{M}_1$
standing	PST	2	
walking	PWA	3	$\mathbf{M}_2$
climbing stairs	PSU	4	
standing in bus	PBS	5	$\mathbf{M}_3$
sitting in bus	PBZ	6	
just holding	HHO	7	$\mathbf{M}_4$
talking on phone	HTA	8	
lying	PLY	9	$\mathbf{M}_5$
cycling	PCY	10	
lying on table	TLY	11	$\mathbf{M}_6$
typing message	HTM	12	

Context Class	No.	Mod.
sitting	1	$\mathbf{M}_1$
standing	2	
lying	3	
walking	4	$\mathbf{M}_2$
cycling	6	
climbing stairs	5	
standing in bus	7	$\mathbf{M}_3$
sitting in bus	8	
table	9	
just holding	10	$\mathbf{M}_4$
talking on phone	11	
typing text message	12	

Context Class	No.	Mod.
sitting	1	$\mathbf{M}_1$
standing	2	
lying	3	
walking	4	
cycling	6	$\mathbf{M}_2$
climbing stairs	5	
standing in bus	7	
sitting in bus	8	
typing message	9	$\mathbf{M}_3$
just holding	10	
talking on phone	11	
lying on table	12	

Context Class	No.	Mod.
sitting	1	$\mathbf{M}_1$
standing	2	
lying	3	
walking	4	
cycling	5	
climbing stairs	6	
just holding	7	$\mathbf{M}_2$
talking on phone	8	
lying on table	9	
standing in bus	10	
sitting in bus	11	
typing message	12	

Table 28: Activity classes per module for varying number of modules.

recognized by one module. This classifier is then divided into two modules, each classifying onto six activities. In the next step three modules classify onto four activities each. Having four modules classifying onto three activities each is the approach, where not only the activities are recognized, but also the *conditional contexts* are reacted on with an individual module. This case is the only one where the grouping of the activities per module is not

arbitrary, but the amount that the module switches are optimized according to the meta knowledge *conditional context*. Furthermore, the complexity and accuracy of activity recognition with six and twelve modules is analyzed, where each module recognizes one or two activity class(es) besides the complementary one.

The data the modules are trained, checked and tested with consists of about 2000 to 5000 feature vectors per activity class. As usual the features are mean and variance over a window of eight samples. The acceleration sensors are sampled with a frequency of about 100Hz, which leads to 12.5 feature vectors per second. The training data for each of the activities consists of 1000 feature vectors per class and the check data of 200. The training data for training the classifier modules on the *complementary class* consists also of 1000 feature vectors randomly selected from training data of the other modules. The same applies for the check data where there are 200 feature vectors. The rest of the data is used to analyze the modular classifiers for accuracy and mean evaluated rules, which indicates the complexity of the dynamic queue classification.

Evaluation Results The results of the evaluation are shown in table 29. Here the different scenarios list the results for accuracy and complexity in rows and columns . Since the clustering in the proposed novel training algorithm is done on each of the class specific training data separately, the resulting rules can clearly be assigned to each of the activities it should map onto. For each of the activity classes, which are identified through the unique char triple, the amount of rules are listed. Also, the number of rules for classifying on the *complementary class* per module can be seen in table 29. For the evaluation data, which is the randomized (50 feature vector slices) rest after cutting of training and check data, the mean evaluated number of rules is calculated. For this, the amount of rules per used module for classifying each feature vector is measured and summed. Each of the sums for classifying each

Description	**# Rules per Activity Class**												**# Rules per *complementary class***						**Accuracy**		**Mean**
	PSI	PST	PLY	PWA	PSU	PCY	PBS	PBZ	HHO	HTA	HTM	TLY	M_1	M_2	M_3	M_4	M_5	M_6	$\tau = .0$	$\tau = .9$	**Rule Ev.**
1 module	1	1	2	7	3	2	3	3	1	1	2	1	-	-	-	-	-	-	68.5	77.2	27.00
2 modules	1	1	1	4	4	2	1	2	2	1	2	2	3	4	-	-	-	-	81.5	92.8	15.47
3 modules	1	2	2	4	3	2	2	2	1	2	2	1	5	5	5	-	-	-	77.7	95.8	15.52
4 modules	1	2	2	4	2	2	1	3	2	1	2	1	5	4	7	5	-	-	83.4	95.3	11.72
6 modules	1	3	6	4	4	2	1	1	2	2	2	1	8	7	6	4	4	7	86.1	98.0	12.36
12 modules	1	1	1	6	2	1	1	2	2	2	2	1	5	4	3	3	4	5	90.4	97.4	7.42
													M_7	M_8	M_9	M_{10}	M_{11}	M_{12}			
													4	5	4	5	4	6			

Table 29: Rules, accuracy and complexity of different numbers of modules.

feature vector are added to the number of all classifications and divided by the total number of feature vectors in the evaluation data set. The resulting number indicates the average complexity of the respective classifier queue for the evaluation data set. This is only a snapshot, but gives an idea about how complex the respective indeterministic activity recognition is. In reality slightly different numbers of average rule evaluations can occur, especially when data should be classified, which regularly needs the calculation of all modules in the queue. This would only be the case if a dynamic classifier queue is running, which can not classify the data properly.

As expected, the worst performance has the monolithic activity classifier with only 68.5% without filtering and 77.2% accuracy with filtered classification on threshold $\tau = 0.9$. Compared to the results of the initial experiments described above, this is still a good result. The increased accuracy of the monolithic approach for the recognition of twelve classes compared to the recognition of nine classes before must be due to the other improvements made to the classifier. The recurrence of the mapping function not only stabilizes the classification, but also enables the improvements due to the filtering. This was analyzed before on the challenge **robustness** in subsection 4.3. Another improvement to the activity recognition compared to the system in the initial experiments is the covariant membership function (eqn. 10) of the RFIS mapping function. Those cover the highly correlated sensor data more accurately and therefore improve the mapping accuracy, which was previously analyzed in subsection 5.1.1. Also, the data to test the approach is focused on the the challenge resources, which does not demand a variant behavior of many users, so it would reflect a real life usage of the mobile phone.

The complexity of the monolithic approach is on the other hand nearly double as high as for all the modular settings. Since the whole classifier has to be calculated on every incoming feature vector, all of its 27 rules need to be evaluated. Compared to the other settings, the highest effort takes the recognition of the activity *walking* with three more rules than average.

The next setting of having two modules in a dynamic queue classifying onto six activity classes each improves the accuracy of recognition significantly. Before filtering the percentage of correct classified activities is increased by 13*pp* up to 81.5% and after filtering by 15.6*pp* up to 92.8*pp*. Also, the average amount of evaluated rules has dropped to 15.47, which are 11.53 rules less evaluated in average compared to the monolithic approach. This reduces the average processor load for the activity recognition to nearly a half. Therefore the energy consumption is also indirectly reduced.

For the next setting of three modules the improvements are not as distinct as before, since the accuracy after filtering could only be increased by 3*pp* up to 95.8%. Before filtering the accuracy has even dropped to 77.7% compared to the setting with two modules. Also, the average amount of rules has increased slightly by 0.05 to 15.52 average evaluated rules for the test data set. Here the overhead for the *complementary class* and the module switch could have caused this effect. A better and not arbitrary grouping of the activity classes per module could negate this effect, which could be achieved through a model driven approach as described and discussed in paragraph 4.4.1. Since the model driven approach is out of the focus of this thesis, the choice in this case was made by the developer.

The next two settings bring improvement to the accuracy of classification and average amount of evaluated rules. The approach with six modules is slightly better in accuracy than the approach with four modules, but the average number of evaluated rules is slightly higher for the setting with six modules. The difference is marginal for both the accuracy and average rules evaluation. A recognition rate of above 95% correct classifications is impressive.

The best performance surprisingly has an activity recognition system with twelve modules with only one class being recognized per module. The accuracy of the dynamic queue of twelve modules without filtering is with 90.4% the highest in this evaluation. In this case the positive effects of the filtering, which increases the accuracy up to 97.4%, could be outweighted by the negative ones. The kind of negative effects the filtering could have were analyzed in subsection 4.6.1. For the average number of evaluated rules of the last setting with twelve modules is with 7.42 is by far the best. The *most* modular architecture of an activity recognition for twelve activity classes is 3.64 times better than the monolithic approach. This fact shows the superiority of the modular approach to cope with the challenge of **resources**. Compared to the system of choice, where four modules are used to detect the twelve activities and to additionaly react on four *conditional contexts*, the last setting is 1.58 times more efficient and 2.1*pp* more accurate after filtering. It seems that the positive effects of the subdivision of a classifier outweighs the overhead due to the *complementary class* and the module switch. Since the test data is randomized according to slices of 50 feature vectors, the sequence of activities does not reflect the sequence in a real life data set. Also, the size of the slices of 50 data pairs (4 sec.) is not unrealistically high, but also not very challenging for a dynamic queue with such a high number of modules. Therefore, in a more challenging scenario the overhead could outweigh the positive effects and the approach with four modules could then be better.

Calculation Effort Since the amount of operations per classification of one feature vector of dimensionality n and m rules was deducted before, the average amount of evaluated rules leads to the numbers of floating point operations listed in table 30. The number of divisions, case switches and maximum calculations is only a rough

Evaluation Setting	$+$	$\times$	$\div$	max	switch	e^x
1 module	4951	5328	6	13	61	27
2 modules	2863.6	3062.12	$\sim$9	$\sim$7	$\sim$31	15.47
3 modules	2868.6	3069.92	$\sim$9	$\sim$5	$\sim$21	15.52
4 modules	2182.6	2324.12	$\sim$9	$\sim$4	$\sim$16	11.72
6 modules	2295.8	2448.56	$\sim$9	$\sim$3	$\sim$11	12.36
12 modules	1504.6	1479.32	$\sim$9	$\sim$2	$\sim$6	7.42

Table 30: Rules, accuracy and complexity of different numbers of modules.

estimation, since these numbers depend on the average amount of calculated modules, which were not measured so far. Also, the non integer of rule evaluation numbers lead to non integer operation numbers, which are only statistical estimations according to the measurements on a test data set.

The derivation of the actual processing time for a certain processor architecture is hardly possible, since other factors need to be taken into account, such as the programming language, the compiler, the indeterministic operating

system scheduling the partly indeterministic processor architecture. In the end the respective activity recognition system has to be tested online on the device in a long term evaluation. The performance can be estimated though, since the number of additions, multiplications and exponential function calculations indicate the superiority of the modular approach.

For a activity recognition on a state of the art mobile phone the calculation time would anyway only be a fraction as on the OpenMoko Freerunner, since the devices have only one accelerometer and usually a floating point unit, which is missing in the evaluation phone.

5.4.3 Energy Consumption - OpenMoko Freerunner

Since the evaluation of the modular activity recognition is based only on offline analysis so far, the next step in evaluation on the challenge **resources** includes measurements on a mobile phone device. The most interesting question to answer is how much energy is consumed by the activity recognition. This kind of measurements need to be done on the mobile phone and can only roughly be estimated on the base of offline data. Additionally, the energy consumption of the activity recognition with a previously described very simple sleep-time scheduling is measured. This is because of the assumption that the energy consumption of the phone in an idle mode is not significantly increased, but the desired much lower suspend mode power drain can not be reached with an *always on* activity recognition. The evaluation setting and results had been previously presented in [28].

In this subsection again a OpenMoko Freerunner phone is used, which is equipped with two 3-D accelerometer sensors, a 400MHz ARM processor (no floating-point unit), a WiFi 802.11 b/g transceiver and a 1200mAh Li-Ion battery. The evaluation includes measurements of the battery capacity and classification results of the activity recognition with and without sleep-time scheduling. To have an upper limit of what the idle mode is consuming, battery measurements were also taken without running the activity recognition.

Also, the percentage of detected activity events is measured. The concept of an activity event was introduced in subsection 4.6.1, because the sleep-time scheduling and the filtering have effects on the activity event detection rate. An activity event in this manner is the period, where only one activity is performed. The activity event starts when the first classification should recognize a new activity and ends when the last classification of this event is done before a new activity starts.

Evaluation Setting To have comparable results for an activity recognition with and without sleep time scheduling, a data replay method was implemented. Also, with the capability of replaying a data set, several intervals of activity class changes could be tested. This is necessary for two reasons: (1) the dependability of the sleep time scheduling on the frequency of class changes and (2) the interference of the sleep time scheduling with the recurrence of the classifier. Since the acceleration sensor and its sampling through the controller needs energy too, the replay mechanism intervenes after the sensor measurement and before the feature extraction. Here, instead of the current acceleration measurement the recorded measurement is inserted.

For evaluation a data set of ~ 72 minutes for nine different activity classes of one person is used. The activity classes which were used are typical to mobile phones. The activity classes are grouped together in the different modules according to the **conditionality**, which is called *conditional context*. All the activities, modules and *conditional contexts* are displayed in table 31.

Besides the replay method, there are some other issues in the evaluation method. One is, that the distribution of the operation system (Debian GNU/Linux neo 2.6.32v20) currently running on the evaluation device does not allow sensor measurements, calculations or active SSH sessions when the phone's display is switched off. To make a remote evaluation possible, in which a normal keyboard could be used and the results could be displayed on a normal sized screen, the activity recognizer is run remotely over a SSH session.

Since the sleep time scheduling results in smaller to bigger leaps forward in time, one question to be answered in the evaluation is if activity events could be detected or not. Furthermore, with the filtering on the reliability measure, which improves the overall recognition accuracy, the amount of classifications are reduced. Here, a correctly detected activity event could be filtered out and therefore the threshold determines how many activity events could be detected. This is the case for both methods of activity recognition, the sleep time scheduled and the one where only the modules are scheduled in a dynamic queue. To especially evaluate this circumstance, the replayed test data is randomized according to slices of 80, 240 and 400 data pairs of sensor measurements, which leads to 10, 30 and 50 classifications (8 sample window size) of the same activity in a row. Here an integer multiple

Conditional Context	Context Class	Class No.	Classifier Module
Phone in users trouser pocket: *no movement*	user is sitting	1	$\mathbf{M}_1$
	user is standing	2	
	user is lying	3	
movement	user is walking	4	$\mathbf{M}_2$
	user is climbing stairs	5	
	user is cycling	6	
Phone in users hand:	just holding	7	$\mathbf{M}_3$
	talking on phone	8	
	typing text message	9	

Table 31: Conditional contexts, classes and classifier modules for the acceleration sensor.

of the window size for feature extraction was chosen, since no window should include data of two different activities. In average there are 5.75 classifications per second, which results in activity periods of ~ 1.74, ~ 5.22 and ~ 8.7 seconds.

When measuring the power consumption there has to be dealt with imprecise information which is provided by the operation system of the device. The OS only provides integer measurements of the percentage of remaining battery power. Here a more precise measurement of the remaining mAh would be desirable. Nevertheless, it can be shown, that the sleep-time scheduled modular classification lowers the energy consumption of the activity recognition, but the *always on* recognition does also not limit the battery runtime significantly.

Activity Recognition without Sleep-Time Scheduling First, the accuracy and energy consumption of the activity recognition without sleep time scheduling and only scheduling of the modules is analyzed. For the percentage of correct classifications the classification without the sleep-time scheduling is the upper limit. The sleep-time scheduled classification can only be as good as this upper limit. For the measurement of power consumption the regular modular activity recognition is the lower limit, where the recognition where also the sleep-time is scheduled can only be better.

Since the sequence size of the activity events has no impact on the energy consumption of the activity classification with only module scheduling, the same remaining battery capacity of 74% for all three trials is measured. Compared to the measurements of power consumption (80% battery power remaining) when no activity recognition is done, the regular modular activity recognition consumes 6% more capacity.

The question is now, how many activity events could be detected and what is the accuracy of the recognition? Since the recognition results of the activity recognition for the event period length with 30 classifications in a row is with 78.2% not very high, a filtering of the classifications on the reliability measure is done. But this filter reduces the amount of classifications passed through to the next level of processing or the application. The tradoff here is how many activity events could be detected.

For a threshold of $\tau = 0.45$ the confusion matrix for an event period length of 30 classifications is shown in table 32 on the left side. Most of the classes have reasonable recognition rates of close to or above 90%. Other classes could be recognized with over 80% accuracy, which is also a accaptable rate. Only one class (*sitting*, class no.1) has low recognition rates with only 63.5% correct classifications. As can be seen, the activity *holding* (class no.7) strongly interferes with the class *sitting* (class no.1), which is due to the very similar position the phone has in the evaluation users pants pocket compared to holding the phone in the hand. This two classes are hardly separable with the available sensor patterns and therefore, the classifier decides for one or the other class. In this case the class *holding* is favored. To have a comparison to the classification results with a higher threshold of $\tau = 0.8$ the confusion matrix is shown in table 32 on the right side. Here nearly all classes could be recognized with over 90% accuracy.

Activity Recognition with Sleep-Time Scheduling Due to the structure of the activity classifier, the upper limit of an recognizer with the regular modular classifier can not be reached through a recognizer with also sleep-time

	1	2	3	4	5	6	7	8	9
1	63.5	1.9	0.6	2.5	2.0	1.3	10.7	0.4	3.1
2	1.5	82.1	0.6	0.5	1.1	0.2	1.0	1.1	1.4
3	0.0	0.4	92.9	0.1	0.0	0.0	0.0	0.0	0.0
4	0.0	1.3	0.2	91.3	5.4	2.0	0.1	1.3	0.3
5	1.3	0.8	0.5	3.4	89.0	1.5	0.1	0.3	0.7
6	0.0	0.0	0.0	1.0	0.2	93.6	0.0	0.0	0.0
7	32.2	6.4	1.1	1.0	1.6	1.0	86.5	0.3	1.9
8	1.1	5.9	4.0	0.1	0.5	0.3	1.3	96.7	1.2
9	0.3	1.2	0.0	0.0	0.2	0.1	0.2	0.0	91.4
	86.6	85.6	59.2	72.2	79.9	90.3	85.0	96.7	83.4

	1	2	3	4	5	6	7	8	9
1	68.0	0.7	0.0	1.9	1.5	0.7	5.9	0.2	3.0
2	1.1	89.3	0.5	0.2	0.9	0.0	0.4	0.6	0.2
3	0.0	0.0	96.8	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.2	0.0	94.7	3.3	0.6	0.2	0.5	0.0
5	0.9	0.7	0.4	1.7	93.5	0.8	0.0	0.1	1.2
6	0.0	0.0	0.0	0.7	0.0	97.3	0.0	0.0	0.0
7	28.9	5.7	0.4	0.5	0.4	0.4	92.3	0.1	1.6
8	0.9	3.0	2.0	0.2	0.4	0.1	1.0	98.5	0.7
9	0.2	0.4	0.0	0.0	0.0	0.0	0.2	0.0	93.3
	51.9	69.1	53.1	31.0	41.1	80.7	45.7	91.9	38.4

Table 32: Confusion matrices for **not sleep-time** scheduled activity recognition for filter threshold $\tau = 0.45$ with 87.5% (left) and $\tau = 0.8$ with 91.5% (right) overall classification accuracy.

scheduling. This is because the modular classification includes a recurrent edge, where the output of the classification at time t is fed back at $t+1$. With this feedback a high recognition rate, a robust classification and the gain of a reliability measure is possible, but the recurrence interferes with the sleep-time scheduling. Here the circumstance with a sleep-time scheduled classifier the last classification of time t that is fed back at time $t+n$ is not always the previous pattern nor activity. This is especially the case when the classification at t is originating in one activity and the sleep-time scheduler activates the recognition at $t+n$ in a different activity.

The remaining battery capacity is, for the trial of a period length of 10 successive classifications, with 74% the lowest. This is the same energy consumption as for a non sleep-time scheduled activity recognition. Here the recurrence stabilizes the recognition mostly at the end of a period, which results in oszilating classifications and therefore nearly no sleep phases. A more precise battery capacity measurement could show marginal improvements.

For the trial of 30 successive classifications periods, the energy consumption of the activity recognition improves by 2% less than without sleep-time scheduling. The third trial with 50 has the highest remaining capacity of 78%, which is only 2*pp* less than without any activity recognition at all.

Again, for the second trial (20 classifications), the confusion matrices for filter thresholds of $\tau = 0.45$ (left) and $\tau = 0.8$ (right) are shown in table 33. The results are lower than with a regular modular activity recognition, but

	1	2	3	4	5	6	7	8	9
1	57.9	1.8	2.2	7.5	6.7	5.6	21.9	0.9	8.4
2	6.7	77.9	0.5	2.8	1.6	0.0	1.9	1.2	1.0
3	0.0	0.3	88.1	0.0	0.0	0.0	0.0	0.0	0.0
4	0.4	3.8	1.1	77.6	12.4	2.1	0.0	1.5	0.0
5	0.8	1.8	2.2	9.3	73.3	1.7	0.4	0.0	0.7
6	0.0	0.0	0.0	0.6	0.6	88.2	0.0	0.0	0.0
7	33.5	6.8	2.2	1.9	3.8	1.4	73.6	0.3	2.7
8	0.8	6.2	3.8	0.3	1.6	1.0	1.9	96.1	1.0
9	0.0	1.5	0.0	0.0	0.0	0.0	0.4	0.0	86.2
	71.3	79.6	51.5	67.2	64.8	79.5	75.5	93.5	75.0

	1	2	3	4	5	6	7	8	9
1	54.7	0.4	1.4	7.0	6.7	2.6	10.5	0.9	9.6
2	2.7	88.4	0.7	0.0	1.3	0.0	0.8	0.6	1.8
3	0.0	0.0	91.8	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.8	0.0	86.0	10.7	0.4	0.0	0.3	0.0
5	1.4	0.4	1.4	4.7	74.5	0.9	0.0	0.0	0.0
6	0.0	0.0	0.0	0.8	0.7	94.9	0.0	0.0	0.0
7	39.9	5.6	2.1	0.8	4.0	0.9	85.5	0.0	3.5
8	1.4	3.2	2.7	0.8	2.0	0.4	2.4	98.1	0.9
9	0.0	1.2	0.0	0.0	0.0	0.0	0.8	0.0	84.2
	41.6	58.7	40.7	27.0	30.7	64.8	35.3	89.6	28.8

Table 33: Confusion matrices for **sleep-time** scheduled activity recognition for filter threshold $\tau = 0.45$ with 79.9% (left) and $\tau = 0.8$ with 84.2% (right) overall classification accuracy.

with a filter threshold of $\tau = 0.8$ many classes can be recognized with over 80% accuracy or more.

Evaluation Summary and Discussion In the evaluation a direct correlation and therefore a tradeoff between classification accuracy and activity event detection percentage could be found. A sleep-time scheduling even tightens this problem, since due to the recurrence of the classifier, the recognition rates are lowered, and the scheduling

reduces the classification periods to detect the events. Also, the filtering, used to eliminate the unreliable classifications and therefore further improving the recognition accuracy, lowers the amount of detected activity events again. The tradeoff, which was analyzed in detail in subsection 4.6.1, for the usage of the sleep-time scheduling is therefore the accuracy of activity event detection and recognition against the power consumption of the activity recognition. All evaluation results are shown together in table 34.

seq. size	τ	sleep-time scheduling			module scheduling			no recog.
		% det. seq.	% class. acc.	% rem. cap.	% det. seq.	% class. acc.	% rem. cap.	% rem. capacity
10	0	65.3	57.8	74	79.4	62.7	74	80
	0.45	59.8	70.3		75.7	74.6		
	0.8	47.6	75.7		63.8	79.8		
30	0	82.6	68.8	76	88.0	78.2		
	0.45	80.7	79.9		87.4	87.5		
	0.8	70.5	84.2		85.0	91.5		
50	0	83.8	70.1	78	86.3	80.0		
	0.45	82.1	83.1		86.0	89.1		
	0.8	75.4	87.7		85.2	92.4		

Table 34: Comparison of results for module scheduled, module and sleep-time scheduled and no activity recognition. Table includes results and measurements for different activity sequence sizes and filter thresholds of percentage of detected activity events, correct detected activities and remaining battery capacity.

Especially the measurements of remaining battery capacity are too coarsely grained to make general assumptions about the sleep-time scheduling algorithm. It is expected, that even for the smallest event size of 10 successive classifications, the sleep-time scheduled activity recognition is lower in energy consumption. Also, the evaluation setting where its not possible to use the phones sleep cycles distorts the results. If it would be possible to use the phones standby mode in between the sleep-time scheduled activity recognition, the power consumption could significantly be lowered compared to an *always on* recognition. At this point it was only possible to evaluate the power savings for less CPU usage due to the sleep-time scheduling, whereas the lion's share would be the utilization of the suspend mode of the mobile device. This is clearly an aspect to be validated in future work.

5.5 Evaluation: Challenge 5 - Conditionality

The last challenge to be evaluated in this thesis is the **conditionality**. Since the user can be situated in differently, can carry the phone on various body positions or the device can be somewhere in the environment, the activity recognition has to deal with a high complexity of sensor patterns which need to be classified. The modularity of the classification process again offers the key innovation to deal with this problem, since each module can cope with a different **conditionality**.

In this subsection the initial experiments to cope with the challenge **conditionality** are presented. Three different system architectures are compared, a monolithic and two modular ones. The monolithic classifier is considered to be the lower limit, where the modular approaches can only be better. The methods are tested on a challenging scenario with six *conditional contexts*, but in the future systems coping with many more **conditionalities** and even different device orientations will be investigated. This evaluation an be seen as initial to have a general understanding about the possibilities of a modular activity recognition to deal with the challenge **conditionality**.

5.5.1 Evaluation

For evaluation upon the challenge **conditionality** three approaches are compared, one monolithic and two modular classifiers. The monolithic approach is expected to be the worst performing one in calculation complexity, as well as classification accuracy of both, *conditional contexts* and activities. One of the modular classification is a dynamic queue of classifier modules which are semantically grouped and the modules classify on one to three activity classes each. This approach is expected to be the best performing one in classification accuracy of the *conditional contexts*. The other modular classification uses modules, where each module is only classifying on one activity class besides the *complementary* one. As earlier experiments had shown, this kind of modular classification has the lowest calculation effort and one of the highest accuracies for activity recognition.

Evaluation Setting As data source the OpenMoko Freerunner phone again was used. Also the feature extraction and the sampling rate remains the same as usual. The sensor data is nearly the same as the one used for the evaluation of challenge extendability of subsection **??**, except that it was extended by data of another user. In this manner the recognition results are not directly comparable to the ones presented with the evaluation of the extensibility.

There are three cases to be evaluated: (1) the activity classes are sorted into modules according to their meta semantic **conditionality** which is called *conditional context*, (2) a monolithic classifier is used to classify on all activities and (3) each module is classifying only on one activity besides the complementary class. The combinations of classes per module, the *conditional contexts* and the activity identifiers for each of the three cases are listed in table 35. There are three *conditional contexts* about the location of the phone to be distinguished: (1) phone is lying on a table, (2) phone is in users trouser pocket, and (3) phone is in users hand. Furthermore four *conditional contexts* of the user can be distinguished: (1) user activity is static, (2) user activity is dynamic, (3) user is riding a bus and (4) user is dancing. The actual recognized *conditional contexts* are the following six combinations of the users and phones location and context: **(1)** phone is lying on table, **(2)** phone is in users trouser pocket and users activity is static, **(3)** phone is in users trouser pocket and users activity is dynamic, **(4)** user holds the phone in her hand, **(5)** phone is in users trouser pocket and user is riding the bus, **(6)** phone is in users trouser pocket and users activity is dancing. So, additional to the 15 activity classes the modular classification recognizes also six *conditional contexts*.

The activity classes and the *conditional contexts* are uniquely identified by a char triple. The first char is identifying the location of the phone, such as pocket (P), table (T) or hand (H). The latter two characters are a combination of the users activity and *conditional context*, e.g. users activity is *sitting*, which is a *static* activity (SI). The experiments in this chapter have shown, that only one char for the location of the phone and two for activity and further *conditional contexts* is not enough when a clear human readable description of the activity and *conditional context* needs to be made. Since during the construction phase of the modular activity recognition architecture this problem was not foreseen and the evaluation did not necessarily demand an extension, the usual char triple is used. For future use especially in a service (sec. 6) with much higher numbers of activities and *conditional contexts*, the extension to a identifier with four or five chars could be necessary.

The data set that the evaluation is based on consists of 2000 to 4000 feature vectors per activity, which makes about 50000 in total. With a rate of about 12.5 feature vectors per second the evaluation data has a length of over an hour. This evaluation data is split into three subsets of training, check and test data. The training data is used

Conditional Context	Context Class	Class No.	Char triple	Modules - Scenario 3	Modules - Scenario 1	Module - Scenario 2
Phone on table:	no movement	1	TLY	$\mathbf{M}_1$	$\mathbf{M}_1$	
Phone in users trouser pocket: *no movement*	user is sitting	2	PSI	$\mathbf{M}_2$	$\mathbf{M}_2$	
	user is standing	3	PST	$\mathbf{M}_3$		
	user is lying	4	PLY	$\mathbf{M}_4$		
Phone in users trouser pocket: *movement*	user is walking	5	PWA	$\mathbf{M}_5$	$\mathbf{M}_3$	
	user is cycling	6	PCY	$\mathbf{M}_6$		
	user is climbing stairs	7	PSU	$\mathbf{M}_7$		
Phone in users hand:	just holding	8	HHO	$\mathbf{M}_8$	$\mathbf{M}_4$	**M**
	talking on phone	9	HTA	$\mathbf{M}_9$		
	typing text message	10	HTM	$\mathbf{M}_{10}$		
Phone in pocket: *user in bus*	user is sitting in bus	11	PBZ	$\mathbf{M}_{11}$	$\mathbf{M}_5$	
	user is standing in bus	12	PBS	$\mathbf{M}_{12}$		
Phone in users: trouser pocket: *dancing*	user is dancing (style 1)	13	PDA	$\mathbf{M}_{13}$	$\mathbf{M}_6$	
	user is dancing (style 2)	14	PD2	$\mathbf{M}_{14}$		
	user is dancing (style 3)	15	PD3	$\mathbf{M}_{15}$		

Table 35: Activity classes, *conditional contexts* and classifier modules for modular activity recognition.

to train the classifier modules. For each of the activity classes, the training data consisted of 1200 feature vectors resulting in 18000 in total. The check data is used to check the success of the training process. This subset consists of 200 feature vectors per activity class and 3000 in total for all 15 activities. The rest of the data was used to test the approach. The test data is varying from 600 to 2600 feature vectors per activity class. The varying amount of feature vectors to test the classification on each activity does not influence the overall recognition rates, since the total percentage of correct classified activities and *conditional contexts* is derived from the individual percentage. Therefore, all classes have the same influence on the overall accuracy.

Accuracy vs. Filter Threshold Three different classification systems are tested first on the average accuracy for the activity and *conditional context* recognition depending on the filter threshold τ. The results are plotted in figure 43. The monolithic classifier (black curve, $+$ marker) is performing with the worst accuracy for classifying the activities. The recognition rates for detecting the activities does not overcome the 55% mark and is therefore

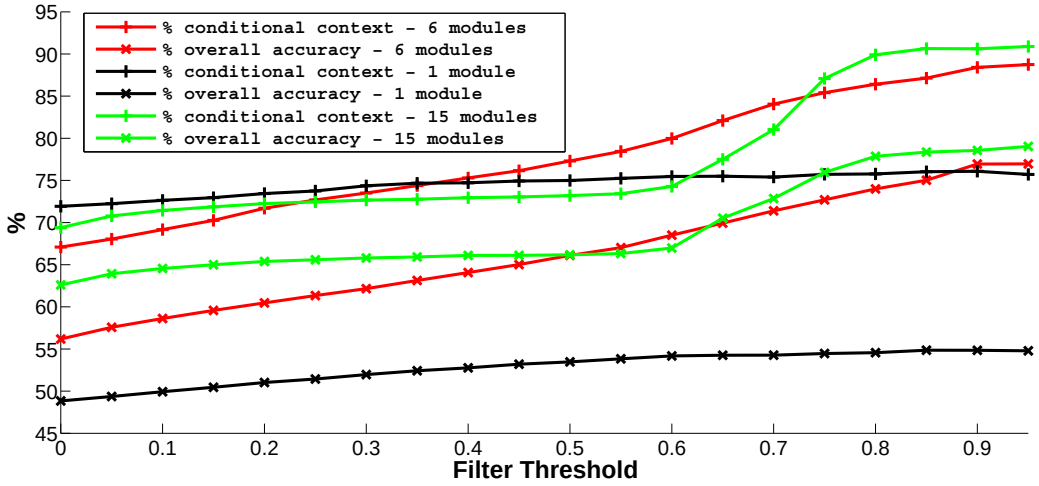


Figure 43: Comparison of three activity recognition systems according to filter threshold, accuracy of activity and *conditional context* detection.

in any case not applicable in a real world setting. Also, the recognition rates for the *conditional contexts* (black curve, x marker) are outperformed by the two modular classifiers with rising filter threshold. For both, the activity and *conditional context* recognition, the accuracy of the monolithic classification can hardly be improved due to a increased filter threshold τ. The curve for the recognition rates of the *conditional context* is even dropping after an initial increase. These results indicate that the monolithic classifier is, despite of the improvments due to the recurrent mapping function and the covariant membership functions, not capable of classifing onto this high number of 15 classes and 6 *conditional contexts*. This confirms the assumption that a monolithic classifier approach is not able to classify sensor data with a high amount of **conditionalities**. Also, the complexity of the classifier is with 26 rules the highest in all three test cases.

Better is the performance of the two modular classifiers, where the approach with six modules (red curves) is generally outperformed by the activity recognition with 15 modules (green curves). The recognition rates for the activity classification with six modules is only in two cases better than with 15 modules, for a threshold $\tau = 0.55$ and $\tau = 0.6$. The accuracy for the six modules classifier is rising nearly linearly, where the classifier with 15 modules has strange s-curves for both accuracies of activity (x marker) and *conditional context* ($+$ marker) classification. The percentage of correct *conditional context* classifications is for the six modules in the range of $\tau = 0.25$ to $\tau = 0.7$ better than the recognition with 15 modules. The difference of both approaches are only marginal in both recognition rates of activities and *conditional contexts* with thresholds $\tau \geq 0.9$. In these cases the two approaches are nearly performing equally well. Also, the calculation complexity of the modular classifiers on the test data set is similar, where the six module recognition needs the average evaluation of 8.36 rules and the 15 modules classifier 6.05.

Detailed Result Analysis The overall accuracies for the recognition of activities and *conditional contexts* only give a general overview, but the detailed look into the confusion matrices of the three test cases indicate how good the classifications are for the respective activities.

	98.6	**97.8**			**73.0**			**90.2**			**95.2**		**75.9**		
	$\mathbf{M_1}$	$\mathbf{M_2}$			$\mathbf{M_3}$			$\mathbf{M_4}$			$\mathbf{M_5}$		$\mathbf{M_6}$		
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	98.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	5.3	0.0	0.0	0.0	0.0	0.0	0.0
2	0.9	95.6	8.7	8.5	2.2	1.4	1.2	0.0	5.7	4.5	0.1	0.0	1.5	1.8	5.3
3	0.0	1.6	89.9	0.8	0.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	88.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0
5	0.5	0.4	0.1	0.5	43.2	3.6	10.8	0.0	1.2	0.2	0.0	0.3	5.9	3.6	0.0
6	0.0	0.2	0.0	1.5	19.2	82.7	31.7	0.0	0.4	0.0	0.0	0.0	5.6	0.9	26.3
7	0.0	0.0	0.0	0.0	3.2	0.0	24.6	0.0	0.0	0.0	0.0	0.0	3.7	0.9	0.0
8	0.0	0.0	0.0	0.0	0.3	0.0	0.0	96.1	0.0	1.0	0.0	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	80.8	0.3	0.0	0.0	0.0	0.0	0.0
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	92.2	0.0	0.0	0.0	0.0	0.0
11	0.0	1.0	0.1	0.0	0.0	0.7	0.0	0.3	0.0	0.0	95.0	0.4	0.0	17.0	0.0
12	0.0	0.0	0.5	0.0	0.0	0.0	0.0	1.8	0.4	0.0	0.1	94.5	0.0	0.0	0.0
13	0.0	0.8	0.6	0.5	30.9	9.4	31.7	0.8	6.1	1.4	0.3	3.8	83.0	8.9	10.5
14	0.0	0.4	0.0	0.0	0.3	2.2	0.0	1.0	0.0	0.2	1.3	0.7	0.4	63.4	31.6
15	0.0	0.0	0.0	0.0	0.3	0.0	0.0	0.0	0.0	0.2	3.2	0.0	0.0	3.6	26.3
	31.0	29.3	52.0	43.9	7.5	22.5	9.0	73.1	42.6	47.7	42.4	45.6	15.9	6.1	1.2

Table 36: Confusion matrix of classifier queue with six modules and filter threshold $\tau = 0.9$. Overall accuracy: 76.9%; *Conditional Context*: CC=88.4%; Average Evaluated Rules: 8.36.

The confusion matrix of the activity recognition with six modules in a dynamic queue and a filter threshold of $\tau = 0.9$ is listed in table 36. The overall recognition of the activities is not very high with 76.9%, but the six *conditional contexts* could be recognized with 88.4% accuracy. Also, six of the activities could be recognized with over 90% and another five with over 80% correct classifications. The lowest accuracy is reached with the

classification of the activities *walking* (class no. 5), *climbing stairs* (no. 7), *dancing syle 2* (no. 14) and *style 3* (no. 15). Since many of the incorrect classifications of class no. 5 and 7 are mutually misclassified or on the activity *cycling* (class no. 6) which all have the same **conditionality**, the recognition rates of the *conditional context phone is in users trouser pocket and users activity is dynamic* is not significantly lowered. Also, a lot of the misclassifications of the classes no. 5 and 7 are made on the activity class *dancing style 1*. This class is strongly overlapping with the normal dynamic activities of the user. Since this activities are on different modules and have different **conditionalities**, the effect can not be compensated in the recognition of the *conditional contexts*. Here the user has to decide whether the recognition of the one or the other activity and *conditional context* is important or she can live with the accuracy of the system.

	87.2	**97.5**			**97.6**			**99.9**			**99.4**		**62.1**		
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	87.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	97.6	0.1	0.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0
3	0.0	0.0	97.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.9	0.0	0.0	96.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.9	0.0	0.0
6	0.0	0.4	0.0	1.8	0.2	99.2	0.0	0.0	0.0	0.0	0.0	0.4	0.0	0.0	3.0
7	0.0	0.0	0.3	0.4	97.6	0.0	96.0	0.0	0.0	0.3	0.0	0.0	87.7	0.0	0.0
8	0.0	0.0	0.0	0.0	0.8	0.0	0.0	100.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0	0.0	0.0	0.0	0.0	0.0	0.0
10	11.1	1.6	0.9	0.0	0.0	0.0	0.0	0.0	0.0	99.7	0.0	0.0	0.0	0.0	0.0
11	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	99.3	0.0	0.0	3.8	8.1
12	0.9	0.3	0.9	0.0	0.2	0.6	4.0	0.0	0.0	0.0	0.0	99.4	1.9	2.1	3.0
13	0.0	0.0	0.0	0.0	1.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	6.5	0.0	0.0
14	0.0	0.0	0.0	0.6	0.3	0.2	0.0	0.0	0.0	0.0	0.5	0.0	0.0	91.9	78.4
15	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.0	2.1	7.6
	16.7	59.9	77.2	49.2	15.8	40.3	10.8	82.4	3.7	52.6	78.4	66.6	9.2	31.7	24.5

Table 37: Confusion matrix of classifier queue with 15 modules and filter threshold $\tau = 0.9$. Overall accuracy: 78.6%; *Conditional Context*: CC=90.6%; Average Evaluated Rules: 6.05.

The recognition rates with 15 modules are listed in the confusion matrix in table 37. The overall accuracy of recognizing the activities is with 1.7*pp* and the *conditional contexts* with 2.2*pp* higher than a recognition with six modules. But with a closer look into the classification results, one can recognize, that nearly all classifications of the activity class *walking* (no. 5) are misclassified on class *climbing stairs* (no. 7). Since these two classes imply the same *conditional context* the recognition rate of this is not affected. But the accuracy for recognizing the *conditional context phone in pocket and user dancing* is greatly affected by nearly all classifications of class no. 13 being misclassified on class no. 7. Here the recognition rate of the *conditional context* is very low with 62.1%. Also, the classes no. 13 and no. 15 with under 10% correct classifications are nearly not being recognized.

Are these two approaches compared directly on the class dependent recognition rates, the one with six modules has a better distribution of the percentages. The accuracy for the *conditional context phone is in users trouser pocket and users activity is dynamic* is recognized with lower accuracy than with 15 modules, but *phone is in users trouser pocket and users activity is dancing* is recognized with higher applicable accuracy. For this *conditional context* the 15 modules classifier is with 62% accuracy are nearly not applicable. Also, the recognition rates for the activity classes is with the six modules classifier more distributed, where the 15 modules approach has recognition rates below 10% for some classes and the activity *walking* can not be recognized at all.

For completeness the confusion matrix of the monolithic classifier with a filter threshold $\tau = 0.9$ is listed in table 38. Here the recognition rates for most classes are below usability. Also, the *conditional contexts* three and six are recognized with 52% and 63.1%, which is significantly lower than the modular classifications and far from applicable. Therefore, this monolithic approach is not usable due to the recognition accuracy of activities, but also the *conditional contexts*. Additionally, the calculation complexity of the monolithic classifier is with 26 rules to be

	81.7	**93.6**			**63.1**			**92.4**			**73.6**		**52.0**		
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	81.7	0.0	0.0	0.0	0.0	0.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	90.9	0.6	0.0	0.0	2.2	0.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	2.9	2.6	86.5	0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.9	0.0	0.0	0.0
4	1.0	1.5	1.4	96.9	2.4	0.7	0.0	0.0	0.0	0.0	0.0	0.5	0.0	0.6	0.0
5	1.0	0.5	1.4	0.5	16.5	16.2	1.7	0.0	0.0	0.0	0.0	0.0	1.6	0.0	1.8
6	1.9	1.1	2.3	0.2	26.6	57.4	13.6	0.0	0.8	0.9	0.0	0.9	5.5	0.6	4.2
7	7.7	1.2	3.2	0.2	28.5	12.5	16.4	3.2	1.5	4.3	0.0	0.9	9.3	1.2	1.8
8	2.9	1.2	2.3	0.5	12.0	6.6	16.4	89.8	0.8	29.6	0.0	0.0	19.8	3.0	1.8
9	1.0	0.2	1.7	0.8	5.7	0.7	23.7	3.8	90.2	30.4	0.0	5.5	20.3	3.6	3.6
10	0.0	0.5	0.6	0.5	5.0	0.7	19.8	2.5	1.5	28.7	0.3	15.1	22.5	1.8	0.6
11	0.0	0.2	0.0	0.0	1.9	1.5	4.5	0.6	3.0	6.1	61.1	28.3	15.9	4.2	2.4
12	0.0	0.0	0.0	0.0	1.0	0.0	3.4	0.0	2.3	0.0	19.0	38.8	3.3	6.5	7.8
13	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	11.7	9.1	1.1	11.9	10.2
14	0.0	0.0	0.0	0.0	0.2	0.7	0.0	0.0	0.0	0.0	7.5	0.0	0.5	57.7	56.6
15	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.3	0.0	0.0	8.9	9.0

Table 38: Confusion matrix for monolithic classifier with filter threshold $\tau = 0.9$. Overall accuracy: 54.8%; *Conditional Context*: CC=76.1%; Average Evaluated Rules: 26.

evaluated more than three times worse than the next best modular approach.

Summary and Discussion In summary, both modular classifiers perform evenly well for this complex scenario with 15 activity classes and six *conditional contexts* for two different users, where the monolithic approach is not applicable. As discussed before, both modular approaches have certain strengths. The approach where the classes are grouped according to their *conditional context* offers the opportunity to more easily compose classifier queues according to the user's usual behavior. Also, as discovered in this evaluation the approach, with six modules it behaves more linear according to the variation of the filter threshold than the one with 15 modules. Here the decision for a certain filter threshold according to the users or applications needs is easier.

The method of classifying on only one class per module offers the most **flexibility**, since the meta knowledge *conditional context* needs not to be present when training the modules. Also, the calculation complexity is the lowest and the accuracy for both the activity and the *conditional context* recognition, is the highest.

Nevertheless, even in this very challenging evaluation scenario the modular reaction onto the different **conditionalities** and the recognition of six *conditional contexts* is not only possible, but also a high recognition accuracy is reached with low calculation complexity. This argues for the usage of modular classifiers, to cope with the challenge **conditionality**.

5.6 Evaluation: Tradeoffs

Last but not least two of the tradeoffs, which are exemplary and can be judged with hard numbers, are evaluated. Other tradeoffs occurred during the analysis and evaluation of the five challenges **flexibility**, **extensibility**, **robustness**, **resources** and **conditionality**, but those have been considered to not easily being analyzable with real data. Therefore, the two tradeoffs which are evaluated in this subsection are exemplary for all the others which occurred or are thinkable.

First, the tradeoff between activity classification accuracy and the activity event detection rate is evaluated. For the modular activity recognition the filter threshold is successively increased, where the accuracy for activity and event detection are compared. The best tradeoff can be found where both rates are equally high.

Last, the tradeoff between the challenges **robustness** and **resources** is evaluated. The **robustness** is accomplished due to the recurrence of the mapping function, but this feedback is interfering with a previously introduced sleep-time scheduling. This sleep-time scheduling of the modular activity recognition is reducing the energy **resource** consumption. The tradeoff between the solutions for these two challenges rely on more than one factor, some of which are not influenceable.

5.6.1 Activity Classification Accuracy vs. Activity Event Detection Rate

To improve the overall statistical accuracy, the filtering upon the reliability measure was introduced. This filtering reduces the amount of classifications which are passed to the next instance of processing. The frequency is indeterministic, meaning that there are periods where nearly no classification is filtered out, where in other cases none are passing at all. This filtering could therefore result in the missing of activity events.

Modular Activity Recognition To determine the tradeoff between classification accuracy and activity event detection rate, the evaluation setting (subsec. 5.4.3) of the challenge resources is reused. There nine activities and three *conditional contexts* were recognized by a dynamic classifier queue of three modules. The evaluation was done upon the accelerometer sensor measurements provided by a OpenMoko Freerunner phone. The evaluation data set consisted of 72 minutes of activities, about 8 minutes and 2760 feature vectors per activity. The classification was done on the mobile phone with a special replay method, since the original setting is used to evaluate the energy consumption of the activity recognition.

To evaluate the influence of different activity event period lengths, three trials are conducted with period length of 80, 240 and 400 successive classifications. With a classification rate of 5.75 classifications per second in average the events are ~ 1.74, ~ 5.22 and ~ 8.7 seconds in length. For the three trials the percentage of detected activity events is plotted against the accuracy of classification in figure 44.

An activity event is detected if in a period of data pairs of one activity class at least one classification is the activity that actually happened. As suspected, the higher the filter threshold is, the less activity events could be detected. Here a tradeoff between classification accuracy and percentage of detected events has to be found, which can be the threshold where the two graphs intersect. But with different event periods, there are different intersection points, so the threshold $\tau = 0.45$ of intersection for the smallest event period of 10 possible successive classifications is chosen. This is the worst case scenario and for normal human behavior the lower limit.

Modular Activity Recognition with Sleep-Time Scheduling Due to a sleep-time scheduling the accuracy of the recognition and the event detection rate are lowered, but the scheduling can reduce the energy consumption significantly (subsec. 5.4.3). This is again a tradeoff, but this time between three goals: accuracy, event detection and energy consumption. The problem can be defused somewhat with a more sophisticated sleep-time scheduling, which helps overcome the issue with the recurrence of the mapping functions. Also, a more realistic activity event period has a better outcome for both the accuracy and event detection even with sleep-time scheduling. The tradeoff between the challenges **robustness**, which includes the activity recognition accuracy, and **resources**, especially the energy consumption, is analyzed in detail in subsection 4.6.2 and evaluated in the following subsection.

The plots of the detected activity events and the accuracy of classification are shown in figure 45, again for different event periods and filter thresholds. As mentioned before, the graphs are generally lower than without sleep-time scheduling, but are similar in shape. The worst results are for the smallest event period of 10 successive

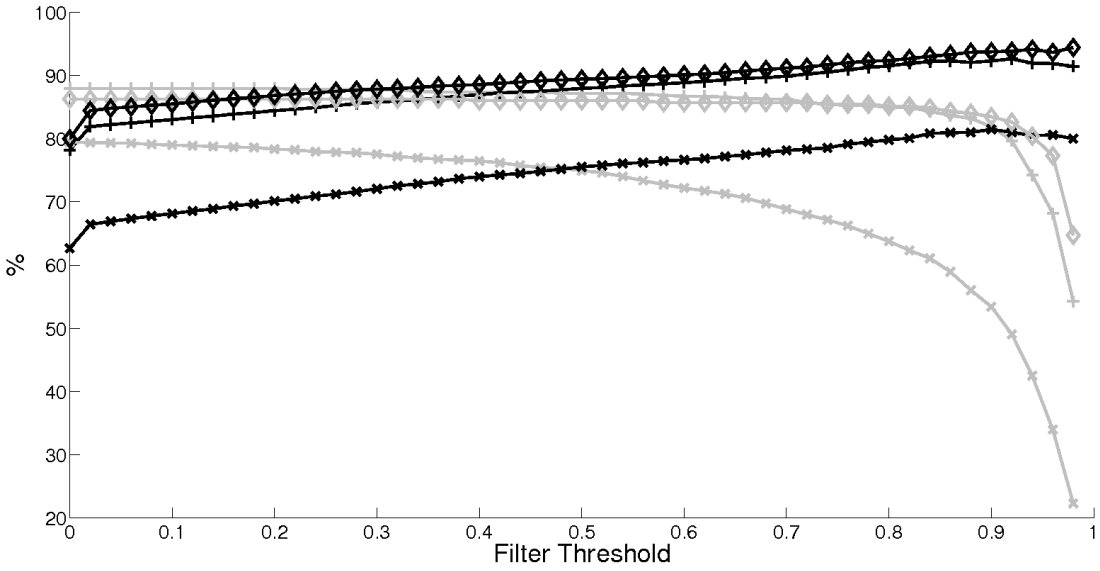


Figure 44: Graphs for percentage of successfully detected activity events (grey lines) and correct classifications (black lines) for different filter thresholds $\tau = 0, .., 0.98$ of not sleep-time scheduled activity recognition. The different sequence periods for equal activities are 10 (x marker), 30 (+ marker) and 50 ($\diamond$ marker) possible classifications.

classifications, where with a maximum sleep period for the sleep-time scheduled classification, no two classifications are of the same activity.

The best tradoff for a activity recognition with sleep-time scheduling can again be determined according to the worst case scenario with a event period length of 1.74 seconds. In this scenario the graphs intersect at $\sim \tau = 0.14$. Since this is the worst case which nearly never happens, because the human behavior is estimated to be much slower, the intersection threshold $\sim \tau = 0.46$ of the next case of period length ~ 5.22 would be taken. In the end the user has to decide according to the application the activity recognition is used by.

Summary and Discussion As shown in this subsection, there is a tradeoff between the activity recognition accuracy and the activity event detection rate. The higher the filter threshold upon the reliability measure the higher the average statistical accuracy of the activity recognition, but the lower the event detection rate. Both accuracy and event detection are lowered due to a sleep-time scheduling, which further reduces the energy consumption of the activity recognition.

A good tradeoff was found for the threshold where the accuracy for the accuracy and event detection is equal, but this threshold strongly depends on the activity event period length. Since the period length of a real life usage of a mobile phone strongly depends on the user, their conditions and the activity itself, it is not predictable nor can a general boundary be defined. Therefore, three different period lengths were used for evaluation, one very short, one average and one longer period. The very short period was used to determine a lower limit. The longest period length of the evaluation can exceed the normal behavior for e.g. the activities *walking* or *standing*, but undercut activities e.g. *sitting* or *lying*. Therefore, most of the cases occurring in real life should be covered through the evaluation.

5.6.2 Robustness vs. Resources

The trade-off between **robustness** and **resources** can not as easily be described with clear numbers as the tradeoff evaluated in the last subsection. This tradeoff is depending on two main variables, the maximum sleep-time length and the activity event period length. The first variable can be influenced, where the latter one depends on the users behavior and is therefore not influential due to any factor of system design.

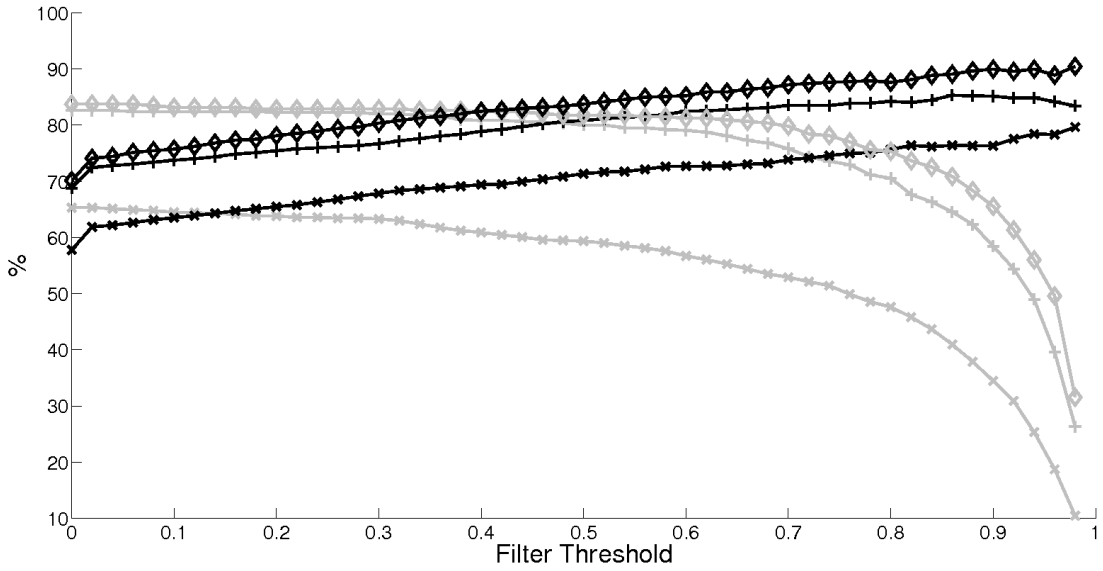


Figure 45: Graphs for percentage of successfully detected activity events (grey lines) and correct classifications (black lines) for different filter thresholds $\tau = 0, .., 0.98$ of scheduled activity recognition. The different sequence periods for equal activities are 10 (x marker), 30 (+ marker) and 50 ($\diamond$ marker) possible classifications.

Evaluation For evaluation of the trade-off between **robustness** and **resources** the same setting as in subsection 5.4.3 is used. There the OpenMoko Freerunner is used to evaluate the energy consumption of a recognition of nine activities with and without sleep-time scheduling. The evaluation uses a replay method, so in the two cases the results are based on the exact same data set. A sleep-time scheduling is used as described in subsection 4.4.2 with a

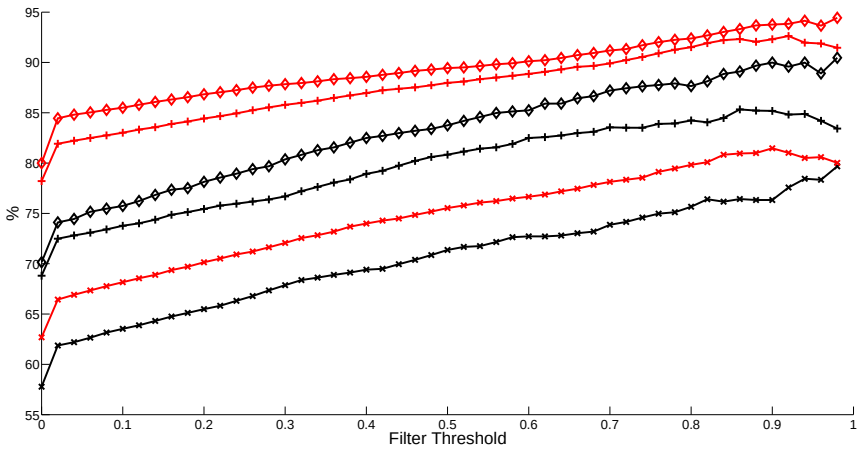


Figure 46: Graphs for percentage of correct classifications of activity recognition without (red lines) and with (black lines) sleep-time scheduling for different filter thresholds $\tau = 0, .., 0.98$. The different sequence periods for the activity events are 10 ($times$ marker), 30 ($+$ marker) and 50 ($\diamond$ marker) possible classifications.

maximum sleep phase of 10 successive classifications. The tradeoff is evaluated with three different activity event period lengths of 10, 30 and 50 successive classifications. The one with 10 classifications is the most challenging for both, the always on activity recognition and the one with sleep-time scheduling. This is due to the fact, that the maximum sleep phase length is 10 classifications and the most incorrect classifications are made when the user has changed the activity.

The results of these three test cases on the mobile phone are plotted in figure 46. There the red graphs are for the

activity recognition without sleep-time scheduling and the black ones with sleep-time scheduling. As can be seen, the recognition without scheduling is always better than the one with, which was expected. Also, the recognition for a activity event period length of 10 classifications has always the worst accuracy even without scheduling. The results for the period lengths of 30 and 50 classifications are nearly the same, where the difference between the two activity recognition systems are more clear.

Summary and Discussion The results are summarized in table 39. The average downgrade for a activity recognition is listed with the maximum and the minimum. The most degradation in average due to the scheduling was

evaluation setting	**average distance**	**min. distance**	**min. threshold τ**	**max. distance**	**max. threshold τ**
event width 10 class.	4.2*pp*	0.3*pp*	0.98	5.1*pp*	0.90
event width 30 class.	7.9*pp*	6.3*pp*	0.70	9.4*pp*	0.02
event width 50 class.	6.3*pp*	3.8*pp*	0.96	10.4*pp*	0.04

Table 39: Summary of tradeoff results.

measured for the activity event length of 30 classifications. The maximum on the other hand had occurred for the period length of 50 classifications. For the case of 10 classifications both activity recognitions are performing nearly equally bad, since for this period length the recurrence of the mapping function has the least stabilizing effect. This case is expected to be very rare in real life usage of a mobile phone since the user has to change activities in a frequency of 1.74 seconds.

6 ActiServ - An Activity Recognition Service for Mobile Phones

The key improvement to activity recognition on mobile phones is the modularity of the classification process. This approach offers **flexibility**[1] in classifier module training, combination, exchange and adaption. The modular activity recognition is **extensible**[1] by new modules, so new activities can be recognized without the redesign of the whole classifier. A recurrent mapping in the classifier modules enables the derivation of a reliability measure, which can be used for filtering to improve the overall recognition rates and it stabilizes the recognition process so that a more **robust**[1] classification results. The modular activity recognition needs less **resources**[1] in computation and energy consumption than a monolithic classifier. The **conditionality**[1] of the device or its user needs special considerations, where the modular concept offers the reaction on different *conditional contexts* with fairly high accuracy and low calculation complexity.

Since the focus of this thesis is the common user and not experts, a mechanism is needed to support the proposed modular activity recognition on the users phone according to their needs, behavior and individualities. Here a service is a method to provide activity recognition to the common users mobile phone. Due to a service faulty or badly performing modules can be adapted or exchanged, which offering **flexibility**. The user can **extend** the activity recognition through a service to recognize new activities. A service can control the activity recognition, so the filtering is adjusted according to the users needs to offer full **robustness** of the recognition. Also, a service can control the sleep-time scheduling to most efficiently save battery **resources**. A service can deal with the different and changing **conditionalities** a user or her phone is situated in by providing modules for the different *conditional contexts*.

As introduced in subsection 3.4 along with the modular activity recognition for mobile phones several tools and implementations were made, to support the training and deployment of the classification onto the users device. The usage of these tools and implementations by the common user is not probable nor in most cases possible. Thats why the automated composition of the tools and implementations in a service (fig. 47) is favored, where the user of this service needs no expert knowledge. The user just commits to the service, collects some data and the rest is done by the service. The service to support the activity recognition, which is introduced in this section, is called ActiServ.

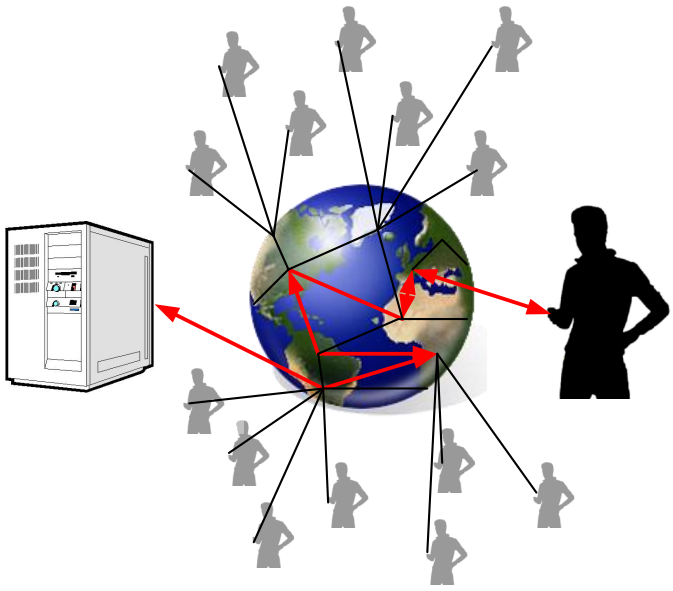


Figure 47: Service to support activity recognition - one for all and all for one.

Furthermore, the supply of annotated sensor data is essential to the training of the modular classifier, but the

[1] The bold words correspond to the five challenges of activity recognition on mobile phones: **flexibility**, **extensibility**, **robustness**, **resources**, and **conditionality**

collection of this data costs the time and effort of the user. To reduce the time and effort the data of all the users of ActiServ is stored in a database and combined on demand. Therefore, one user does not have to collect a sufficient amount of data for the machine learning, but only a minimum to provide relevant information about behavior and conditions. This approach also guarantees that the training data is variant enough and the resulting classifier recognizes activities in more than one position or orientation of the phone. To have only the single user collect sensor data with this kind of variability in a reasonable amount of time is mostly not practical.

The ActiServ provides mechanisms to **extend** an activity recognition so it detects more classes. The user just has to collect some data for this new activities and the ActiServ provides new classifier modules. Here again the combination with data from other users is possible, if others do have the same requests. Also, the users activity recognition system can partly be adapted, exchanged or recombined in a **flexible** way due to ActiServ. When the user discovers that the recognition rates are too low, she can give feedback to ActiServ, which is modifying the classifier queue to reach applicable recognition rates again. An every day usage of a mobile phone will result in changing **conditions** for the activity recognition. The ActiServ provides new or additional modules to cope with changing or reoccurring *conditional contexts*.

In this section first the architecture of the ActiServ is described, which components it uses for what purposes and which workflows it has. The main components are the modular activity recognition itself, a global (GTS) and a personal trainer service (PTS). The PTS serves the single user to react on the users personal needs and to have a personal and most possibly best recognition. The GTS is responsible for training the classifier modules. Therefore, it is constantly running in the background, training new sets of modules on varying combinations of user data. It also interacts with the PTS when e.g. a user demands the immediate training of a activity recognition system. Also, the possibilities of the ActiServ to react on the different challenges[1] are described and discussed.

Last, the ActiServ is briefly evaluated offline with a large data set of 20 different users. The main workflow of downloading the application, collecting some sensor data, classifier selection, activity recognition personalization and classifier training is simulated offline according to actual users data. Since the modular activity recognition is the main innovation presented in this thesis, the service to provide this novel concept to the common user is only briefly evaluated. Other workflows, components and mechanisms of ActiServ will be evaluated in future work. The architecture of the ActiServ and the basic workflow evaluation was mostly published before in [25].

6.1 The Architecture

As the common user has no knowledge of machine learning nor of pattern recognition, classification or any details of activity recognition, the support for an activity aware mobile phone due to a service must be extensive. The user does only have to download the activity recognition application and collect some sensor data. The **flexibility**, **extensibility** and **conditionality** can only be coped with due to a service. The other challenges **robustness** and **resources** need at certain times a controlling or an intervention due to a service, too.

This proposed service-based approach to support activity recognition on mobile phones is called ActiServ, whose architecture is now briefly introduced. As the ActiServ is not in the main focus of this thesis, the described architecture is far from extensive nor tested and evaluated in an online setting. Also, the service described here is focused on one scenario, where the user downloads the activity recognition application, than collects some data for a given set of activities and from this point on the ActiServ takes over. This is assumed to be the main thread of ActiServ which is analyzed in the evaluation. Additionally the extension, adaption or exchange of the dynamic queue or the classifier modules are briefly defined, but not evaluated, since the general solutions due to the modular activity to this challenges **flexibility**, **extensibility** and **conditionality** were already analyzed in detail.

6.1.1 General Overview of the Service Components

The service consists of several steps of activity classifier training, optimal classifier selection, classifier personalization, and user accuracy feedback. Each step is realized by different components either located on a server or the user's phone. Mostly after the download of the activity recognition application, the user has to collect a small amount of activity data which the service uses to select an initial recognizer. This activity recognizer is then delivered to the user's device. In the next step the recognizer is personalized, which improves the recognition rates. This step requires more time for calculation, but is still very responsive as will be seen later. The data provided by the user is used in a recurring training process, further improving recognition. Based on the user feedback, the server processes can further improve the activity recognition for the specific user as well as the global quality of the service for the community as a whole. Each step can be repeated at any time and certain services for further improving activity recognition are constantly running, making it possible to achieve recognition rates near 100% within the community. Also the activity recognition provided by the service is more variant than a recognition, that was solely trained on the respective user. All steps and components of ActiServ are displayed in figure 48 and described below. Subsequently, the workflow of the service is presented.

- **Activity Classification Module Set (ACMS):** The ACMS is the collection of classification modules which are suited for the respective user and are running on the user's mobile phone in a dynamic queue at a given time. Therefore it is responsible for recognizing the users activities. The ACMS is specified in the Classifier Specification Format (CSF), which was defined in detail in subsection 3.4.2.

  The architecture of the ACMS is the key innovation in this thesis and was described in section 3. The whole service architecture is built up to support a user specific and best performing ACMS. All the ACMSs which are running on users devices or are just trained for a possible future use are stored in a database with information about the capabilities and its origin.

- **Global Trainer Service (GTS):** The GTS is the key component which trains new Activity Classification Module Sets (ACMS). For training the ACMSs the algorithm described in subsection 3.3 is utilized. The GTS frequently checks the database for new user activity data sets or for combinations of user data which have not been used. When data is found, the GTS creates a new ACMS using the new data combination. Through the GTS the database is constantly filled with new ACMSs, where badly performing ACMSs are replaced or deleted to ensure that the database always consists of the best performing ones.

  Also, if users want to extend their recognition by new activities, new classifier modules have to be trained, which is also done by the GTS. If more than one user wants to have the same additional activity to be recognized, the GTS proceeds for this subset of users as with the usual ACMSs. This training of additional modules is usually triggered by a PTS instance of the first user which demands the recognition of new activities.

  The PTS also triggers other ACMS or single module training in the GTS when the user feedback is indicating this request. Also, not only one instance of the GTS is running at one point in time, since many ACMSs can be trained in parallel with multi-core servers.

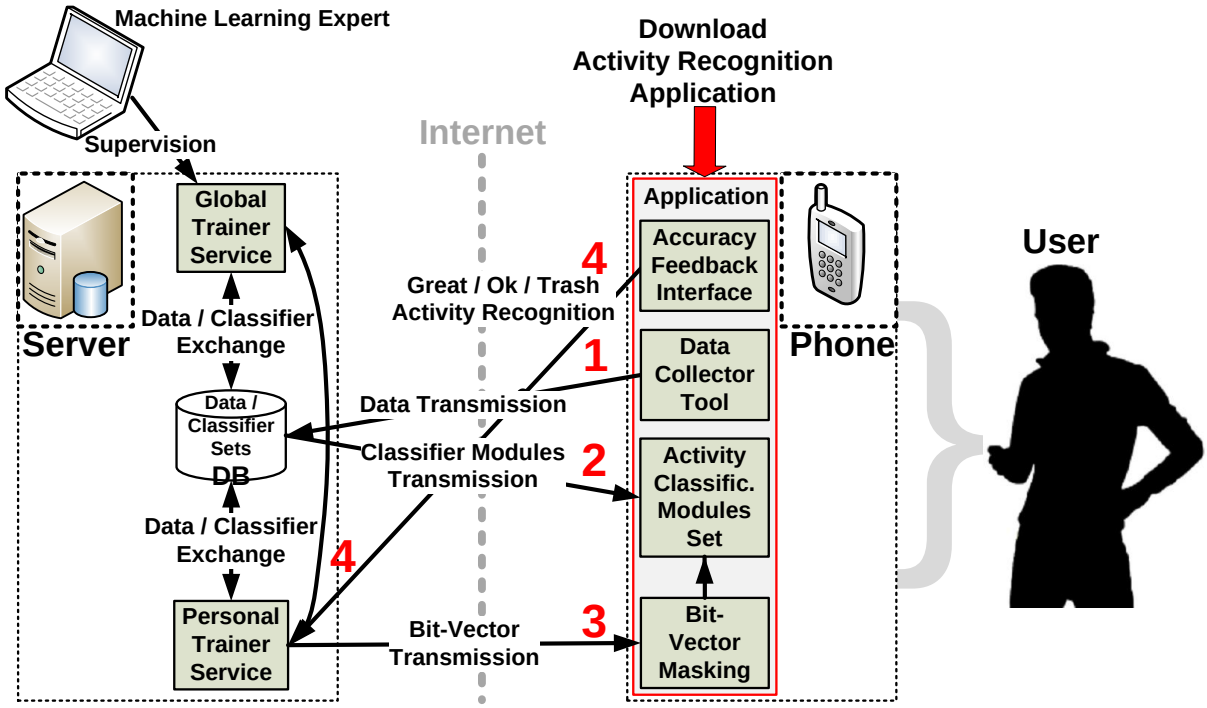


Figure 48: ActiServ service architecture.

- **Personal Trainer Service (PTS):** The PTS selects which ACMS is delivered to the users device. The decision of which ACMS is selected is based on the performance of the classifiers over the annotated data collected by the user. The PTS is also responsible for personalizing the classifier modules, a process accomplished using a genetic algorithm to enable and disable features of the training data via bit-masking. The bit-vector masking was explained in detail in subsection 3.1.6 and the genetic algorithm to search for a suitable bit-vector was explained in subsection 3.3.7.

  PTS also identifies the bit-vector to mask the users ACMS, and therefore personalizes the activity recognition for the respective user. The bit-vector is identified on the data set the user provides through the Data Collector Tool (DCT) via a genetic algorithm. The PTS also can be located on the users mobile phone itself, so new personalization data need not be transmitted to the server. This is currently not implemented due to the efficiency of the personalization process, but will be if certain bottle necks are identified and eliminated.

  The PTS is also responsible for triggering new instances of the GTS, when a user demands additional classifier modules to recognize new activities. In this case the PTS has to identify new bit-vectors for the preexisting modules of the users current ACMS, so the transition onto the new modules can be recognized (subsec. 4.2). Also, if modules need to be adapted or exchanged due to changed user **conditions**, the PTS identifies new bit-vectors, searches in the database for modules which could be used in exchange or triggers the new training of modules in the GTS.

- **Data/Classifier Set Database:** This unit is the central storage for the ACMSs and user data. All the users data is collected in a database located on a server. The data is saved with additional informations about the user, such as the users **conditionalities** and the users mobile phone. Since a ACMS can only run on the mobile phone series it was trained on, the information about the users mobile phone is required. Also, the ACMSs can not unconditionally be trained by combining data from different users **conditions**, like e.g. one user is carrying the phone in the trousers and the other one in the jackets pocket.

  All the trained ACMSs get stored in the database. The ACMSs get stored with additional information about the data they were trained on, which users they are suited for, which **conditionalities** they can cope with and

what mobile phone the recognition works on. This information is on the one side necessary, as the devices brand and series, or useful, as the **conditionalities** so the PTS does not have to look through the whole database to find a fitting ACMS. Also, if bit-vectors are identified for certain ACMSs, these get stored in the database and linked to the users data and the ACMSs.

The transmission of data and ACMS all go through the database. The GTS stores the ACMSs and removes them if the set is replaced by a better performing. PTS selects ACMSs for the respective users from the database and transmits it to the user. The bit-vectors also get saved and linked by the PTS.

- **Bit-Masking:** The bit-masking provides a method for personalizing and adapting an ACMS without destroying its original activity recognition capabilities. The bit-vector masking mechanism was defined in subsection 3.1.6. The personalization is done to adapt an ACMS, which was selected by the PTS from the database according to the performance on the users data, onto the specific sensor patterns of the respective user. The masking is only temporary, so a new bit vector can be provided and used for masking at any time. A new personalization of an already personalized ACMS is therefore possible.

  The bit-vector can also be used to adapt onto changed **conditions** of the user like changed clothes, physical or mental conditions. This capability was analyzed in the evaluation of the challenge **flexibility** in subsection 4.1. Also, the bit-vector masking can be used to enable modules of a running ACMS to recognize transitions onto newly added modules, which was analyzed in subsection of the challenge **extensibility**. Both facts will not further be analyzed in this section.

  The bit-vectors are identified with the algorithm described in subsection 3.3.7 by the PTS. The vectors get transmitted to the user's phone and used to mask the currently running ACMS. Also, the vectors get stored in the database and linked to the users data and the ACMSs they are applicable on.

- **Data Collector Tool (DCT):** The DCT is used by the user on the mobile phone to collect annotated activity data. In the subsection *Tools and Implementations* 3.4 the DCT has already been described in detail for the different platforms. The user selects the activity they want to perform from an extensible set of activities in a drop-down-list, carries out the activity, and then pushes a button to stop recording.

  Every activity for which the user has already collected data is marked, so the user does not twice collect data for the same activity. The user can collect the data for all activities in sequence or whenever the user is in a situation where the user can perform the respective activity. A best ACMS can only be selected when for all initially supported activities data is collected, but a good performing one can already be transmitted if the dataset is not complete yet. This increases the acceptance of the activity recognition, since the user has not to collect the complete dataset at once when she downloads the application, but can collect it gradually whenever she has the time.

  The DCT directly interacts with the database to store the data. When new data is stored a PTS instance for this user is created which than selects a suitable ACMS according to the data. Also, for the ACMS which is transmitted to the users phone, a bit-vector is identified upon the users data to personalize the ACMS.

- **Accuracy Feedback Interface (AFI):** With the AFI, the user can give feedback to the GTS/PTS as to whether their activity recognition is working. This feature has not been implemented nor evaluated so far. The mechanisms to react to the feedback are currently under development.

  Three kinds of feedback messages will be provided, recognition was *well done*, *ok*, or should be thrown into the *trash can* (fig. 49). This feedback will not be used for personalization per se, as the personalization process is done using the user data, but rather could be used to change the training process in general. This is currently not implemented and outside of the scope of this thesis. The feedback could also be used to adapt certain modules onto changed conditions as analyzed in subsection 4.1. For this purpose a more fine grained feedback would be needed.

Since nowadays servers offer multicore capabilities, more than one instance of the GTS could be executed. The PTS needs one instance per user anyway, but each instance is not running permanently. The PTS thread for one user is demanding the most processing when the user submits the collected activity data where it has to select the best ACMS and personalize it. Afterwards the PTS is only activated periodically when new better ACMSs are available or the user has special requests.

Figure 49: A possible GUI for the AFI.

Also, the GTS, the PTS and the database can be located on different servers as they communicate over the network to have the ActiServ as scalable and **flexible** as possible. Different instances of the PTS or GTS could also be executed on different servers if the amount of users demands that.

6.1.2 Workflow

There are different threads of the workflow, since the supply of a ACMS, the maintenance and **extension** are separate aspects of the ActiServ. The main thread is about the supply of a working ACMS to the user. This thread involves the most parts of the application on the users device and the components of the service on the server. The maintenance of a running ACMS does include the **flexible** adaption and exchange of classifier modules, which are faulty or badly performing. The user always can initiate the thread for an **extension** of an ACMSs by new modules, which would recognize additional activities.

Main Thread The main thread of the ActiServ is initiated when the user commits to the service. First, the user downloads the ActiServ components to the phone. Second, a small amount of initial data must be collected, where each of the activities available in the drop down list of the DCT is carried out for 1-2 minutes. This process can be done sequentially or the user starts the collection whenever the user is doing the respective activity anyway. When the data set is not completed on every supported activity, the ActiServ can proceed with this incomplete user informations, but this results in an activity recognition which is performing sub-optimally.

When all or a majority of the activities are completed, the data is transmitted (step **1** fig. 48) to the ActiServ server. On the server the data is saved in the database with additional information like the users id, **conditionalities**, mobile phone brand and series. At this point, the interaction between the user and the service in the main thread is finished and will only be re-initiated over the AFI if necessary. On the server the PTS selects the best performing ACMS from the database based on the collected data and transmits the set in the CSF (subsec. 3.4.2) to the user's device (step **2** Fig.48). This step has nearly no delay, since only a search over the pre-existing ACMSs must be done and no training is performed. According to the additional information provided by the user the search for an ACMS can be accelerated. Here the optional information about the **conditionalities** of the user can provide the most speedup. The information about the users mobile phone device are mandatory, since sensor data of different brands and series can not be mixed. The transmitted ACMS can now run on the users phone and recognize activities with an initial accuracy.

Meanwhile the PTS is identifying the bit-vector to personalization the ACMS which is currently running. This process takes time (depending on efficiency and complexity about 1 hour), but since an ACMS is already running on the user's phone, the delay is not directly recognizable for the user. The personalization data, which is just a bit-vector and not a complete ACMS, is transmitted to the phone in step **3** (Fig.48). On the phone the bit-vector masking component personalizes the ACMS, which can be done during runtime in between classifications. Now

the phone should have reasonable activity recognition rates (see evaluation), but the process runs further on the ActiServ server.

The GTS constantly trains new combinations of ACMSs, in which the new user's data is included as well. Over time, an ACMS is present in the database which has been trained on the data from the new user and can be transmitted to the user's phone (recognition rates can rise to above 97%). This new ACMS can be personalized again via the PTS and therefore can be further improved. The user can also give feedback to the system via the AFI (step **4** fig.48), but this is not necessary. If the ActiServ is in a faulty state due to bad user data, then an expert can intervene.

Thread of Extension of ACMSs If the user wants to have new activities recognized the ActiServ can provide additional classifier modules to the running ACMS. To have the current ACMS modules recognize the transition onto the new modules, the ActiServ also provides bit-vectors to adapt the modules. If the extension does not work properly (the contrary has been was shown in subsec. 4.2), then the ActiServ can also provide an ACMS which was entirely trained on the original and new activities.

The extension starts again by collecting 2-3 minutes of data for the new activities. Here some aspects need to be considered, then the addition of new activities to the DCT would be possible without an active connection to the service, but the combination with other users data of the same activity would in this case not be possible. Therefore, the user needs to be connected to the service, where in a separate list all additional activities would be listed, so if the user can find her desired supplementary activities the combination of user data in the training process is possible again. It is also likely, that already a classifier is trained for these activities. The collected data is again transmitted to the server in step **1** (fig. 48).

Three sub-threads are now possible, one for the case that no other user already supplied data for the additional activity. Then the users PTS instance need to trigger a new thread of the GTS to train one or more new classifier module(s) (depending on the amount of new activities to be recognized) on the data provided by the user. Meanwhile the PTS itself searches for bit-vectors to mask the old modules of the user's current ACMS, so they recognize transitions onto the new ones. After the GTS instance has finished the training the new module(s) get submitted to the user in step **2** (fig. 48). Also, the bit-vectors are transmitted to the user's phone in step **3** (fig. 48) after the PTS has finished identifying them.

The next sub-thread that is possible is for the case, where data for the activities was already collected by other users, but no classifier module has been trained so far or the trained modules do not classify the user's data appropriately. In this case, the PTS needs to trigger the GTS again to train one or more classifier module(s) and identify bit-vectors itself. These get then again transmitted to the user's phone in step **2** and **3** (fig. 48).

The third sub-thread occurs when the user demands additional activities where already classifier modules are trained for and which are suitable for the user. Then the personal instance of the PTS selects the best performing modules according to the user's data from the database and directly submits it to the user's phone in step **2** (fig. 48). It is hardly possible that one of the bit-vectors used for the other users is fitting, so the PTS has to identify new ones instead. After the identification the PTS transmits the bit-vectors to the user's phone in step **3** (fig. 48).

In the end the user can give feedback to the ActiServ about the performance of the new modules and the ACMS in general (step **4** fig. reffig:ActiServ). In case the feedback is negative, new additional modules can be trained or the user needs to collect some data again. The recollection of data is probable if the user is solely demanding the recognition of the new activities. In this case the collection of 2-3 minutes of training data might not be enough.

Thread of Adaption or Exchange of Modules Due to the AFI the user can give feedback to the ActiServ about the activity recognition. If the feedback indicates faulty or bad behavior of one or more modules, then this module(s) can be adapted or exchanged. Therefore, the thread of adaption and exchange usually starts with the last step **4** (fig. 48). There, in contrast to the example in figure 49, the user needs to give detailed feedback that activities are not recognized correctly. From this feedback the service can deduct which module needs to be adapted or exchanged. Then the user again needs to collect some data by performing the respective modules activities for 2-3 minutes. This data than gets transmitted to the ActiServ server (step **1** fig.48).

If the module is identified, then the user's PTS instance is either identifying bit-vectors to adapt the module or initiates an instance of the GTS which trains a new classifier module. In both cases the PTS eventually needs to identify bit-vectors for the other modules, since the interference with the respective module can cause faulty

classification, too. Also, modules for exchange could already be present in the database, where they just need to be selected and combined with bit-vectors for some of the other modules. The process of adaption and exchange of modules was analyzed in detail in subsection 4.1. Therefore, the following four sub-threads are possible:

First, the faulty or badly performing module gets adapted. In this case the personal instance of the PTS is solely responsible for the adaption, by identifying a bit-vector for the respective module and possibly for some of the other modules, too. The bit-vectors then get transmitted to the user's phone (step **3** fig. 48).

Second, the module needs to be exchanged. Here the user's PTS instance triggers a new thread of the GTS, which trains a new module. This then gets transmitted to the phone in step **2** (fig.48). The exchange of a module mostly demands the adaption of other modules of the ACMS, since through the complementary class they are all interconnected. The bit-vectors get identified by the PTS itself and then are transmitted to the user's phone (step **3** fig. 48).

Third, the module needs to be exchanged, but the PTS directly finds a suitable module according to the user's data in the database. This module could have been trained for a user with the same faulty module or the module from a regular ACMS is suitable. In this case the PTS just needs to select the module, transmit it to the user's mobile phone (step **2** fig. 48) and then identify the bit-vectors for some of the other modules in the user's ACMS. The bit-vectors are submitted to the phone in the step **3** (fig.48).

Fourth, all modules of the ACMS are badly performing and the exchange and adaption of some modules would not solve the problem. In this case the main thread, which was explained before, is executed, where the user starts first to collect data for every activity to be recognized.

In the end the user again can give feedback over the AFI to the ActiServ, whether the ACMS is now correctly recognizing the activities performed by the user. If the feedback is negative, the described process starts over.

6.1.3 Global Trainer Service (GTS)

The GTS is responsible for training the ACMSs. The main threads of the GTS are constantly running in the background combining user data and training new ACMSs. Due to the service being available on two main smart phone platforms, the iPhones and Android based phones, a large set of users can participate in the service and contribute data to the community. Therefore, after the availability of ActiServ (still under development) an increasing set of user data will be available for training ACMSs, which will keep the GTS instances constantly busy. Also, the GTS is triggered by the PTS if new ACMSs or single modules need to be trained.

Training Data Complexity A classification system which is trained on a large data set of multiple users has disadvantages. First, the training algorithms have long calculation times, since the training data set increases with every user added to it. Second, the resulting Activity Classification Module Set (ACMS) is inaccurate and complex. Since the diversity of the training data due to the different users is high, the classifier modules need a large number of rules in the RFIS mapping function to map the data. The data is partly contradictory, because different users have different patterns for certain activities. In this case, either one user's training data is preferred to solve the conflict, or both are classified with low accuracy.

The training of a ACMS on only one user will result in either a classifier that is too specific to the training data. This ACMS will only recognize activities which are performed in exactly the same way as the ACMS was trained on. Also, the phone needs to have the same conditions like orientation as during the collection of the training data. A solution to this problem could be to have the user collect a large amount of training data with a high variation in conditions. Since the user does not want to invest a long time for data collection and also has no knowledge about what data could result in a variant classifier [68], the data of many users is combined.

Instead of training the ACMS on the training data of all or only one user in the database, subsets of user-specific data are selected, combined and used to train the classifier modules. Instead of one set of modules, various sets trained on subsets of the users are gained. All the ACMSs are stored in a database, where only the best and fittest ACMSs remain and the worse performing are deleted. With each new user added to the database new data combinations of users become possible, so the GTS is constantly running and training new ACMSs.

Activity Classifier Module Training To train a set of activity classifier modules (ACMS), the machine learning algorithm described in detail in subsection 3.3 and published in [26] and [28] is used. This algorithm is fast,

compared e.g. to the ANFIS [81] methods, and is especially suited for highly correlated sensor values as provided by 3-axis accelerometers. Also, the algorithm provides a **robust** classifier due to the recurrence. Furthermore, it requires only a minimal amount of supervision from an expert, since most parameters are standard and need no alteration. The part of each activity classification module that needs to be trained is the RFIS mapping function.

The training algorithm runs for a preset amount of iterations, where the first iteration is resulting in a non-recurrent mapping function and those that follow are recurrent ones. Since the number of iterations determine the runtime of the algorithm, the number should not be too high. The large amount of experiments conducted with the algorithms indicated that a maximum occurs within the first ten iterations. An actual deployment of the service onto a large set of users phones could result in more reliable results.

Another aspect that influences the runtime is the genetic algorithm, which is used in the inner loop of the training algorithm. This genetic algorithm has an indeterministic runtime, since the initialization is random and the space is searched heuristically. Therefore, the runtime can not be determined in advance, but be influenced due to the parameters of the genetic algorithm. Here the stall time limit and the stall generations limit are the influential ones. They stop the algorithm if after a certain time of searching or amount of generations (iterations) the algorithm can not find any better solution to the problem. Also, according to experience, the inner loops can be limited, since the upper limit due to the subtractive clustering mostly constrict the search space significantly.

Training Data - Random Selection The training $\mathcal{V}^{tr}$ and checking data $\mathcal{V}^{ck}$ for one classifier modules set $\mathcal{M}_l = \{\mathbf{M}_{1l}, .., \mathbf{M}_{Nl}\}$ consists of data from randomly selected users ($u_h \in \{u_1, .., u_A\}$), represented by $\mathcal{V}_{u_h}$. The training data $\mathcal{V}^{tr}_{\mathcal{M}_l}$ for the classifier module set $\mathcal{M}_l$ is a subset selection of the data $\mathcal{V}^{tr}_{\mathcal{M}_l} \subseteq \mathcal{V}_{u_g} \cup .. \cup \mathcal{V}_{u_h}$ for $g \neq .. \neq h$, where each of the classes should have equal amount of training data pairs and the data from every user should contribute same amount of data sets. For the check set $\mathcal{V}^{ck}_{\mathcal{M}_l}$ the process is the same.

All user data is saved in a database, from which different combinations of data are randomly selected. On each selection a new set of classifier modules $\mathcal{M}_l$ is trained. The set $\mathcal{M}_l$ is then checked with the respective checking data $\mathcal{V}^{ck}_{\mathcal{M}_l}$ on classification accuracy. This process is repeated until all combinations of user specific data are trained and checked, where only a certain population of ACMSs which have the highest classification accuracy is saved. With new user data coming in through the community, the process can theoretically run indefinitely. The amount of stored classifier module sets should increase with the amount of new user data added to the database. Since storage space is limited, the amount cannot be increased indefinitely.

The number of users whose data is combined in the training data was determined arbitrary in case of the following evaluation. From a theoretical standpoint it is hardly possible to determine the optimal amount of users. Some of the users provide data which is more variant, in which case less users data need to be combined. Other users collect data in only a few conditions and with small variation. In this case the data needs either to be combined with other user data that is more variant or many of the non variant user data needs to be combined. The actual deployment of the ActiServ to a large set of phones could give a better understanding of this parameter.

Training data - Selection According to Conditionalities There are other methods besides just randomly selecting the user specific data from the database, which is combined as training data. One method could be the mathematical overlapping of the data per activity. If the data is overlapping to a too high degree ($\mathcal{V}_{u_i} \cap \mathcal{V}_{u_j} \approx \mathcal{V}_{u_{i \vee j}}$ with $i \neq j$), the resulting classifier could be not variant enough if the single user's data is not variant itself. If there is nearly no overlapping of the user data ($\mathcal{V}_{u_i} \cap \mathcal{V}_{u_j} \approx \emptyset$ with $i \neq j$), then the data could be partly contradictory and the resulting ACMS not functioning at all. Therefore, a combination is searched for which is in between these two extremes ($\mathcal{V}_{u_{i \vee j}} >> \mathcal{V}_{u_i} \cap \mathcal{V}_{u_j} >> \emptyset$ with $i \neq j$). The determination of an exact rule can be done, when the actual deployment of the service provides large sets of users data and their evaluation will show what kind of combinations lead to a good recognition. Here the feedback through the AFI of the users can help deriving rough numbers about the deployed ACMSs.

Another method could be the accession of informations about the **conditionalities** of the users phone usage. The user could provide this information through a form, which the user fills out before submitting the data to the ActiServ. According to this information the ability to combine two users data can be determined. E.g the data of an user who carries the phone upside down in the pants pocket would not be best to combine with data of a user who carries it in the opposite direction. Whereas, two users who carry the phone mostly in the same orientation, the combination could provide the necessary variance but not be contradictory. Again, a better understanding of this

process the real world deployment of the ActiServ should deliver.

In the end the combination of these two methods could deliver the best results. The questionnaire could give a rough estimation to pre group the data, so the overlapping calculations are more resource friendly and faster. The determination can then on the pre grouped datasets give detailed information about good combinations of user data.

Database Pruning As the GTS constantly trains new ACMSs, the database is also constantly increasing in size. Therefore, a mechanism to prune the database is needed. Some of the ACMS are after training not saved in the database by the GTS, since they can not classify the data. In this cases the training process failed for some modules of the ACMS, because for the training data the used numerical instable methods in the training algorithm could not produce a valid RFIS mapping function. The brief evaluation in this section will show that this case is occurring, but not in many ACMS trainings.

Another metric to decide if a ACMS should be saved or deleted by the database pruning of the GTS is the accuracy of the ACMS on the training and checking data. The whole set of an user's data is not used to train the ACMS, but only a part of it. This is especially the case when data of many users is combined, as too much training data will slow down the training process and could again result in numerical destabilization of the training algorithm. In this manner each user's dataset is split into training, checking and testing data. The training data is used to train the ACMS and the checking data to validate the outcome, since the data is independent from the training data. The third fraction is at this point not used at all, but could be applied in further developments of the ActiServ.

The accuracy of the newly trained ACMS on the training and checking data can now indicate, whether it should be saved in the database. If the performance of the ACMS on the training data is high, but on the checking data low, the indication could be that something went wrong in the training phase. Such ACMSs could be over trained, so they perform very well on the training data, but badly on the checking data. This ACMSs would not be saved by the GTS in the database. Also, if the performance of the ACMS on checking and training data is significantly lower than comparable ACMSs, the module set would also not be saved in the database. The comparable aspect would be, that two ACMSs include training data from same users. In the pruning phase of the database by the GTS the deletion of ACMSs with low performance on training and checking data can be done afterwards. This is the case when new more accurate ACMSs are present, which outperform older ones. Therefore, the GTS frequently has to check the database to delete old obsolete ACMSs.

ACMSs which are currently in use on at least one user's mobile phone are locked and excluded from the pruning process. This is so ACMSs which are in use are not deleted and therefore can not be supported by the ActiServ anymore. Their performance on the training and checking data is still used to compare the new ACMSs to. If there is a significantly better performing ACMS for one user than the one currently running on the user's phone, then the GTS would trigger the users PTS instance to exchange the ACMS. Afterwards the old ACMS would be locked and again be included in the pruning process.

An important role in the ACMS ranking is the AFI. The positive feedback through the AFI is, among other things, used to rank the ACMSs in the database. All positive feedback of a user is added to her current ACMS fitness value, the the higher its fitness and the less likely it will be deleted from the database in the future. Also the negative feedback can be used to rank the ACMSs. The more often a user gives bad feedback about an ACMS, the more likely it will be deleted. A more detailed understanding of such an approach can be gained when the actual service is deployed and tested with a large amount of users.

6.1.4 Personal Trainer Service (PTS)

One PTS instance is responsible for a single user. It is responsible to find the best suited ACMS for this user according to the activity data the user provides. It also identifies the bit-vectors for personalization and adaption of single modules and whole classifier queues. Frequently it checks the database for a new better performing ACMS for the user exists to exchange it with the currently running one if necessary. It communicates with the GTS if a new ACMS or modules need to be trained when the recognition rates drop or new activities want to be recognized.

PTS Purpose After the user has collected the data for the supported activities, this data is submitted to the server the ActiServ is running on. A new instance of the PTS is then created for this user, which will serve the users needs until the user leaves the service. The ACMS $\mathcal{M}_l$ which performs best for the current user is selected by

the PTS. This will then be personalized to recognize activities with acceptable accuracy. For personalization a bit-vector masking is used, which does not destroy the original classification capabilities and is easily removable. The identification of the bit-vector is also the responsibility of the PTS. After the bit-vector is saved in the database and transmitted to the user's phone, the PTS is becoming inactive.

From time to time the user's PTS instance awakes and checks the database for a new and better performing ACMS that is available for the user. This is the case when the GTS has trained a ACMS where the user's activity data was included. The user can also trigger the PTS through the AFI, where the user can give positive, neutral or negative feedback. According to negative feedback the PTS has to actively look for better ACMSs in the database and if it cannot find any, trigger the GTS to train one with the user's data included. Also, the PTS can adapt or exchange one ore more module(s) in the user's ACMS if they are badly performing or faulty.

The user also can request new modules for her ACMS, which are capable of recognizing new activities. In this case the PTS searches the database on existing modules which are capable of recognizing this activities. If the PTS succeeds, the modules are transmitted to the user's phone and the PTS is identifying in parallel bit-vectors for the preexisting modules, so they recognize transitions onto the new ones.

ACMS Selection In the main service threads and afterwards when the performance of the ACMS demands that, an ACMS has to be selected from the database by the PTS. The selection is done based on an initial data set which the user has previously collected. The data set should consist of a significant amount of data pairs for all activity classes. For every classifier module set $\mathcal{M}_l$ ($l = 1, .., A$) the classification accuracy for the new user data is calculated. The classifier module set $\mathcal{M}_l$ that has the best classification accuracy is then selected for the activity recognition on the user's device. An exemplary selection process is shown in figure 50.

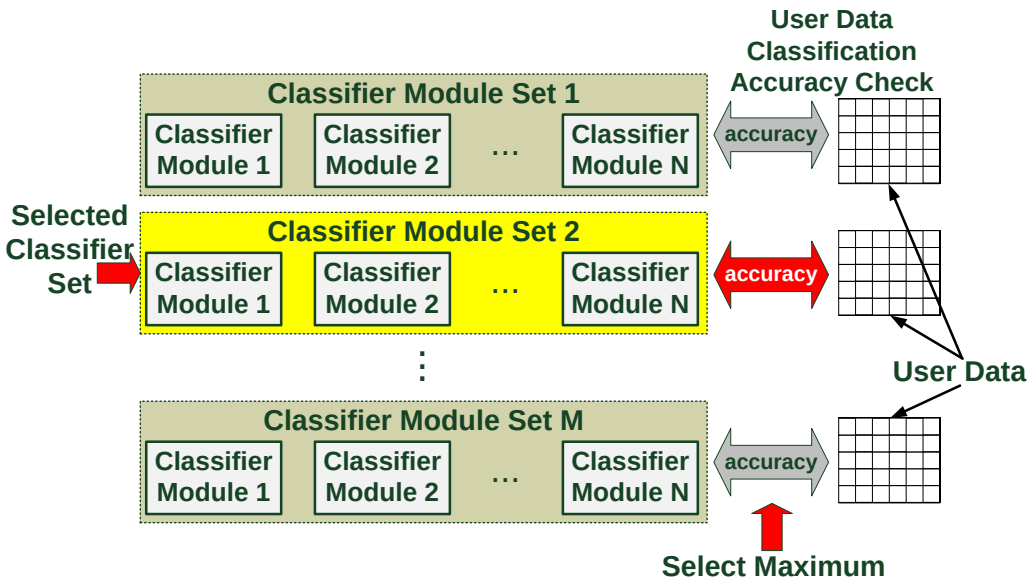


Figure 50: Classifier module set selection.

Two metrics are qualifying for the selection of the respective ACMS, one is the percentage of correct classified feature vectors and the other is the mean squared error (MSE) of the RFIS mapping functions output. Since the MSE does not tell how good the classification actually is, but only how accurate the mapping, the percentage of correct classifications is used to rank the ACMS.

Another method that can additionally to the classification percentage be used on the user's data to rate the ACMS is the feedback through the AFI. As was described with the database pruning, the feedback of the users can be used to apply a fitness value to the ACMSs in the database. This value can also be used when the PTS has to select an ACMS for the respective user. First it would select a group of the best performing ACMSs according to the user's data, then decide according to the fitness value. Therefore, the accuracy of the ACMSs not only on the users

offline data is relevant, but also the actual performance online on the users devices for a real life usage. Detailed knowledge of the usage of such a fitness value can only be conducted with the deployment of the service and is not part of the evaluation presented in this thesis.

Also, the the pre-grouping of suitable ACMSs can be done according to the **conditional** information possibly provided by the user. ACMSs which were not trained on data from equal devices are not usable anyway, but more fine grained information about the locations the phone is carried in or in which orientation can help limit the search space for the PTS significantly. Here again, a real world deployment of the service can give a better understanding about which kind of **conditional** informations are needed, what the user can and wants to supply.

Bit-Vector for ACMS Masking There are several cases where a bit vector masking of certain modules of a ACMS is needed. The most common one is in the main thread of ActiServ, where ACMSs are personalization on the respective users. There the ACMS was just selected from the database by the PTS and submitted to the user's phone. This ACMS needs to be personalized to reach applicable recognition rates. Another case in which a bit-vector masking is needed is, when new modules are added to an ACMS. There the preexisting modules of the ACMS need to be masked, so they recognize transitions onto the new ones. Another case is, when modules need to be adapted or exchanged due to faulty behavior. When a module needs to be exchanged, some of the other modules of the ACMS need to be adapted, so they still recognize transitions onto the exchanged one. The adaption itself is again done with a bit-vector masking. Detailed information about which modules need to be masked in which case can be found in the respective subsections of the challenges **flexibility** and **extensibility**.

The adaption is done via a bit-vector, which specifies the *active* and *inactive* dimensions of each rule for one module $\mathbf{M}_i$. Therefore the bit vector $bit_{\mathbf{M}_i}$ for module $\mathbf{M}_i$, which has n input dimensions and m_i rules, is $b_i := n \cdot m_i$ positions long. To use the bit vector, an interpretation function $I(\mathbf{M}_i(\overrightarrow{v}_t), bit_{\mathbf{M}_i})$ is defined, that *switches* the rule's dimensions temporarily without changing the module $\mathbf{M}_i$ permanently. The interpretation function I is defined as a function mapping a module $\mathbf{M} \in \mathbb{F}(\mathbb{R}^n, \mathbb{R})$ together with a bit vector $bit_{\mathbf{M}} \in \{0,1\}^*$ of appropriate length onto a module $\mathbf{M}'$. The bit-vector masking approach was described in detail in subsection 3.1.6 and partly published in [31].

A genetic algorithm is used to determine the respective classifier module's bit vector. The algorithm to identify a bit-vector was described in subsection 3.3.7. The bit-vector is specified for the respective user according to the data $\mathcal{V}_{u_h}$ on which the module set $\mathcal{M}_l$ was selected. Since the combinations of bits in the bit-vector exclude a full search, a genetic algorithms is used as a heuristic to limit the search space. The space that has to be searched is 2^{b_i}, a complete search would therefore have a runtime of $O(2^{b_i})$, which is impossible to calculate in a reasonable amount of time. According to experience, the genetic algorithm can find a sub-optimal, but appropriate solution in a time span that is acceptable in many applications. Nevertheless, currently methods to limit the search space are investigated. Here, the overlapping of membership functions can indicate which dimensions should be *deactivated*, so overlapping is reduced or even eliminated.

6.2 Offline Activity Classification Evaluation

The ActiServ system was evaluated offline to first have an understanding about its capabilities. The detailed implementation of the service is not done yet. Therefore, the evaluation results of a real world deployment of ActiServ and the online test of the service with a large amount of users is not yet possible. This would also be out of the focus of this thesis, where the main aspect to be analyzed is the novel modular activity recognition itself.

First, the evaluation settings will be explained, followed by a evaluation of an upper approximations for the optimal case, where the training data for the classifier modules are only gathered from the evaluation user. Further, the lower bound approximation is the accuracy of the classifier modules for a data set from the evaluation user, but with a different phone orientation (phone is upside down in the pants pocket).

Second, an ACMS is selected from the database where data from the current user is not present, meaning there is no ACMS available which was directly trained on data from this user. The accuracy of the best performing ACMS under these circumstances is presented. Next, personalization using the PTS to provide bit-vectors for each module is presented, where data from the current user is excluded from the database.

Finally, the performance of the whole system including data from the evaluation user as well as the community is presented, where different phone orientations and personalization are applied. Only the main thread of the ActiServ's workflow is evaluated, since the challenges **flexibility** and **extensibility** have been analyzed before.

6.2.1 Evaluation Setting

As mentioned before, there is a difference between context classes and *conditional contexts*. The classes are the direct output of the classifier, whereas the *conditional contexts* can implicitly be identified through the classifier module which is currently active. The classes are sorted according to semantics of the *conditional contexts*. Three general *conditional contexts* were determined, *the phone is on a table*, *the phone is in the pants pocket* and *the user has the phone in her hand*. Since the amount of classes (6 classes) for the *phone in pants pocket* **conditionality** is too much to be classified with one classifier module, it was decided to use two modules for this state, each classifying onto three classes. These two subset classifiers do not have mutually exclusive classes, but it was decided to sort the classes into subsets according to the amount of movement in the acceleration patterns. The *conditional contexts the phone is in the pants pocket* is split in subsequent contexts according to the amount of movement of the phone, *movement* or *no movement*. Therefor, the *conditional contexts the phone is in the pants pocket* has two more conditionalities, which provide additional information about the user and her mobile phone. The combinations of contextual states, classes and classifiers for the acceleration data classification are shown in table 40.

Conditional Context	Context Class	Class No.	Classifier Module
Phone in users pants pocket: *no movement*	user is sitting	1	$\mathbf{M}_1$
	user is standing	2	
	user is lying	3	
movement	user is walking	4	$\mathbf{M}_2$
	user is climbing stairs	5	
	user is cycling	6	
Phone on table:	no movement	7	$\mathbf{M}_3$
Phone in users hand:	just holding	8	$\mathbf{M}_4$
	talking on phone	9	
	typing text message	10	

Table 40: Conditional contexts, classes and classifier modules for the acc. sensor.

Data from 20 users (16 male, 4 female) age 20 to 32 was collected. All users had experience with using mobile phones. It was tried to ensure that the collection of activities was as realistic as possible. Each test user generated 2-3 minutes of data for each of the ten activity classes during normal everyday activities, resulting in over 500 minutes of data in total.

The classifier module sets are trained on randomly selected sets of four users. 19 of the 20 users were used during the training phase, leaving the data from one user for evaluation of the ActiServ system. The male test subject used for the evaluation provided data with two different orientations for the *pocket* activities. All the test data used in the following evaluation is randomized in slices of 16 data pairs. This is a stress test in the evaluation, since due to the recurrence in the modules, the most false-positives occurred in between activity classes.

In the following tables the recognized activity classes are filtered on a threshold of $\tau = 0.75$. This reduces the amount of classes which are recognized by a certain amount, depending on the general detectability of the respective activity. Which percentage of the classification remain after the filtering is shown in the last row of the confusion matrices. It is expected that a remainder of more than 8% of classifications to be acceptable, so on average still one classification passes through the reliability filter per second.

6.2.2 ACMS Trained on Evaluation User

The best results for activity recognition can be achieved when the ACMS is directly trained on the current user, but the resulting classifier will not be variable in usage. The test subject collected data over 3 minutes per activity class with the Data Collector Tool (DCT). Data collection resulted in about 2250 feature vectors per class and 22500 in total for all ten classes. Out of this annotated data set, 600 samples per class were extracted for training data $\mathcal{V}^{tr}$ and 400 for checking data $\mathcal{V}^{ck}$. For training the complementary class of each of the modules, 1200 data pairs were randomly selected from training data of the other classifier modules and added to the training data. For the check of the complementary class, 800 data pairs were added to the checking data. The remaining data – $\sim$ 1250 annotated feature vectors per class – was used to test the ACMS. The upper limit results for the trained ACMS are presented in table 41 on the left side.

A lower baseline limit can be expressed when the classifier was trained on one phone orientation (e.g. for the phone in the pants pocket) and the evaluation is carried out on data from the user when carrying it in a different position, which represents a major problem in activity recognition. This special case is used for evaluation, because a normal classifier which is only trained on one user's data is incapable of classifying such data, but a classifier which is provided and supported by a service can cope with this problem. The recognition results for the test data with a different phone orientation then the training data are shown in Tab. 41. Here the classes for the *conditional context phone in user's pants pocket* have extremely low recognition rates, where the activities with only one possible orientation still have high accuracy. In the evaluation of service-based approach it will be shown, that the recognition accuracy achieved is not significantly lower than the upper limit and can exceed accuracy for the lower baseline limit.

	$\mathbf{M}_1$ (98.1%)			$\mathbf{M}_2$ (96.4%)			$\mathbf{M}_3$	$\mathbf{M}_4$ (99.3%)		
	1	2	3	4	5	6	7	8	9	10
1	97.8	0.8	0.1	4.2	0.2	0.7	0.4	0.0	0.3	0.0
2	0.6	96.3	1.2	0.9	0.0	0.0	0.0	0.0	0.1	0.0
3	0.0	0.0	97.4	0.0	0.1	0.0	0.0	0.0	0.0	0.0
4	0.8	1.3	1.2	92.3	0.1	0.2	0.0	0.9	0.6	0.0
5	0.0	0.0	0.0	0.3	97.7	1.6	0.0	0.0	0.0	0.0
6	0.0	0.0	0.1	0.0	0.0	97.1	0.0	0.0	0.0	0.0
7	0.5	0.0	0.0	0.0	0.0	0.0	99.5	0.0	0.0	0.0
8	0.3	1.0	0.0	2.2	1.5	0.3	0.1	97.6	1.1	0.0
9	0.1	0.5	0.0	0.2	0.4	0.1	0.0	1.4	97.8	0.8
10	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.1	0.1	99.1
	83.9	83.2	93.5	69.4	57.5	74.5	91.7	86.3	83.2	93.5

	$\mathbf{M}_1$ (64.6%)			$\mathbf{M}_2$ (51.4%)			$\mathbf{M}_3$	$\mathbf{M}_4$ (99.7%)		
	1	2	3	4	5	6	7	8	9	10
1	20.7	44.7	0.1	30.1	10.2	20.0	0.1	0.0	0.1	0.0
2	0.2	29.3	0.7	12.4	2.5	21.8	0.0	0.0	0.1	0.0
3	0.0	0.0	98.2	1.0	0.2	0.0	0.0	0.0	0.0	0.0
4	0.0	20.9	0.5	55.2	84.1	13.5	0.0	0.2	0.5	0.0
5	0.0	0.6	0.2	0.6	0.6	0.3	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0	0.3	0.3	99.8	0.0	0.0	0.0
8	79.2	4.1	0.3	0.7	2.0	40.6	0.1	98.5	0.7	0.1
9	0.0	0.4	0.0	0.0	0.1	3.5	0.0	1.3	98.5	0.6
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.1	99.3
	39.6	22.8	92.1	31.4	34.8	11.9	94.6	89.8	89.8	96.2

Table 41: Upper limit classifier system solely trained on evaluation user (left, conditional context (cc): 98.3%, overall: 97.3%.) and lower limit classifier system for position 2, solely trained on actual user's position 1 (right, cc: 78.9%, overall: 60.0%.) with general filter threshold $\tau = 0.75$.

6.2.3 Evaluation User Excluded from Training

As as before, the upper limit classifier modules could be provided to the user, but since it is not capable of recognizing the activities with the phone in a slightly different position, this approach would make no sense for a real world application. The user could collect data that is variant enough, so the resulting activity recognition could classify more variably. Since the user would need to collect much more data than with the performance of each activity for 2-3 minutes, this approach is not suitable. Also, the user would need to collect data with different positions of the phone, which would presuppose the general understanding of the underlying machine learning process. As initial experiments have shown [68], a common user is not capable of such a data collection.

In this evaluation it will be shown that due to ActiServ the lower baseline limit for a different phone orientation can be exceeded without any previous knowledge of sensor data collection by the user. Also, the time for data collection is limited to 2-3 minutes per activity, which is much less then for a normal machine learning is needed. This evaluation follows along each step of the main thread of the ActiServ.

GTS Trained ACMS for Random Selection of User Data The Global Trainer Service (GTS) has trained 20 Activity Classification Module Sets for randomly selected sets of four users from the 19 users in the database. The training data $\mathcal{V}^{tr}$ consisted of 300 randomly selected feature vectors per class and user, which makes 1200 pairs per class. Again, training data from the other modules was added to train the respective classifier module on the

User Combination	Accuracy (%) $\mathcal{V}^{tr}$	$\mathcal{V}^{ck}$	User Combination	Accuracy (%) $\mathcal{V}^{tr}$	$\mathcal{V}^{ck}$
17, 2, 1, 11	86.0	83.1	18, 4, 7, 15	85.6	81.4
12, 11, 1, 7	83.9	83.2	4, 7, 2, 15	85.06	83.7
6, 14, 15, 16	NaN	NaN	12, 17, 7, 16	NaN	NaN
16, 13, 12, 7	NaN	NaN	9, 13, 18, 17	82.9	82.5
3, 6, 8, 4	85.2	82.7	13, 6, 2, 16	82.7	80.2
15, 16, 12, 2	66.3	66.1	3, 17, 16 14	69.5	64.5
13, 8, 6, 1	84.5	80.2	13, 1, 16, 2	83.9	80.7
10, 7, 12, 2	84.7	81.2	10, 15, 4, 5	84.7	82.8
5, 1, 10, 1	72.5	71.4	7, 5, 17, 12	82.7	80.8
9, 5, 18, 8	82.7	69.0	5, 13, 19, 8	NaN	NaN

Table 42: GTS combinations of user data, accuracy of ACMS (training/checking data). Gray rows indicate faulty ACMSs that are deleted.

complementary class. In total 12000 data pairs for training of all four classifier modules were used. The checking data consisted of 6000 data pairs. The accuracy for the ACMSs, trained on training data $\mathcal{V}^{tr}$ and checking data $\mathcal{V}^{ck}$, is shown in table 42.

For some of the ACMS an error occurred during training and the accuracy on the training and checking data is 'Not a Number (NaN)'. These ACMSs are deleted by the GTS from the database along with badly performing ACMSs. Not all of these ACMS modules are faulty, but the set only works as a whole. Also, some of the worse performing ACMS are deleted by the GTS in further iterations which was explained under the term *pruning*.

Best Selection of Classifiers A new user now demands an ACMS to recognize the 10 activity classes in this evaluation setting. The PTS selects the best performing ACMS on the user data and uploads the ACMS to the user's phone. The best selection is the ACMS trained on the users 10, 7, 12, and 2. The recognition rate without filtering is 56%. The confusion matrix for a filter threshold of $\tau = 0.75$ is shown in table 43 on the left. A mean recognition rate of 62.4% ($\tau = 0.75$) is low, but the more user invariant activity classes were recognized with over 90% accuracy. Also the *conditional contexts* have an accuracy in detection of 71.2%, which is already a good recognition rate, for an ACMS that was provided without delay.

Personalization Data and Test Data Have Same Orientation The PTS now provides a bit-vector for masking the selected ACMS, and therefore personalizes the activity recognition. The resulting confusion matrix is displayed in table 43 on the right. The recognition rates have improved significantly by 23.6 *PP* up to 86%. Still, classes no. 4 and no. 6 have low recognition rates. These classes are mostly misclassified onto class no. 1 of a different classifier

	$\mathbf{M}_1$ (7.7%)			$\mathbf{M}_2$ (79.7%)			$\mathbf{M}_3$	$\mathbf{M}_4$ (98.5%)		
	1	2	3	4	5	6	7	8	9	10
1	0.7	1.1	0.0	7.8	0.9	1.7	0.2	0.2	0.1	0.0
2	0.5	4.6	14.8	0.0	0.3	0.6	0.1	0.0	0.0	0.0
3	0.0	0.2	1.1	0.0	0.0	0.0	0.1	0.0	0.0	0.0
4	62.6	1.9	71.6	44.1	0.9	1.1	0.7	0.9	3.0	0.4
5	1.4	0.2	2.3	0.6	95.8	5.8	0.1	0.0	0.0	0.0
6	33.6	0.0	0.0	0.0	0.6	90.3	0.0	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0	0.0	0.0	98.9	0.0	0.0	0.0
8	1.1	3.5	10.2	38.5	1.5	0.5	0.0	97.8	1.8	0.5
9	0.0	88.5	0.0	9.0	0.0	0.0	0.0	1.1	95.1	3.8
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	95.3
	14.1	37.9	5.6	19.3	12.9	31.3	46.5	20.6	46.1	36.2

	$\mathbf{M}_1$ (90.2%)			$\mathbf{M}_2$ (72.8%)			$\mathbf{M}_3$	$\mathbf{M}_4$ (97.0%)		
	1	2	3	4	5	6	7	8	9	10
1	85.4	2.7	1.0	31.5	1.7	29.8	0.2	0.2	5.9	0.9
2	1.4	84.9	0.4	1.5	0.4	0.9	0.0	0.0	0.8	0.2
3	0.0	0.7	94.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	1.3	9.1	2.4	54.9	0.4	0.0	0.1	0.0	0.8	0.0
5	0.1	1.0	1.3	0.0	92.5	5.5	0.1	0.0	0.0	0.1
6	11.6	0.0	0.0	0.0	3.9	61.3	0.0	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0	0.0	0.0	99.3	0.0	0.0	0.0
8	0.0	1.6	0.1	10.5	1.0	1.7	0.2	98.4	1.2	0.0
9	0.0	0.0	0.4	1.5	0.0	0.9	0.1	1.4	90.9	0.9
10	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.4	97.9
	19.9	24.9	42.5	23.4	18.4	11.6	41.9	22.6	25.9	40.8

Table 43: Best performing selection (left, cc: 71.2%, overall: 62.4%) and best performing selection personalized on new user (right, cc:89.8%,all:86.0%) with filter threshold $\tau = 0.75$.

module. Here the personalization process of the PTS can be improved, so that the classifier module $\mathbf{M}_1$ is not personalized to the point it interrupts the capabilities of module $\mathbf{M}_2$. But this is a trade-off, where the stop criterion of the genetic algorithm used in the PTS plays the key role. The modules either have equal average recognition rates or one is favored to the disadvantage of the other. This can also happen between classes classified through the same module.

Personalized Opposite Orientation in Training and Test Data The PTS has personalized the ACMS via a bit-vector masking according to a data set, where the phone has only one orientation in the user's pants pocket. Now it is investigated what happens if the user is carrying the phone with a different orientation, which is a situation that frequently occurs in daily usage of mobile phones. The results of a test set with about 22000 data pairs for a different phone orientation are shown in Tab. 44. The accuracy is on average low at 63.6%, but as mentioned

	$\mathbf{M}_1$ (79.3%)			$\mathbf{M}_2$ (83.6%)			$\mathbf{M}_3$	$\mathbf{M}_4$ (99.4%)		
	1	2	3	4	5	6	7	8	9	10
1	54.4	87.8	0.6	1.1	0.8	9.0	0.0	0.0	1.4	0.1
2	1.8	1.5	0.6	0.2	0.1	0.1	0.0	0.0	0.2	0.0
3	0.0	0.0	91.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	14.0	5.9	2.3	85.3	77.5	73.3	0.0	0.0	0.0	0.1
5	3.5	0.0	1.0	5.8	8.7	0.1	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0	0.0	0.0	99.9	0.0	0.0	0.0
8	22.8	4.2	1.0	7.4	12.3	13.6	0.0	99.4	1.2	0.0
9	0.0	0.4	3.3	0.1	0.7	3.8	0.0	0.6	96.1	0.5
10	0.0	0.2	0.1	0.0	0.0	0.1	0.0	0.0	1.1	99.3
	1.2	15.3	43.1	43.5	48.0	44.5	94.4	50.0	59.2	82.3

Table 44: Best performing selection personalized on new user, test data has different orientation. τ=0.75, cc: 90.6%, overall: 63.6%.

before, these results need to be compared with the lower baseline approach. There the ACMS was directly trained on the users data with one phone orientation, which is the general approach for a user who has no knowledge of machine learning techniques. Compared to the lower baseline, the recognition rate was exceeded by 3.6 *PP*. With the recognition rate for the *conditional contexts* a more significant improvement by 11.7 *PP* up to 90.6% can be reached. Here user feedback through the AFI can trigger a repeated personalization via the PTS, or the GTS

could have already trained a new ACMS which would perform better with the different phone orientation. Another possibility would be to exchange the modules one and two with modules that are trained on both orientations or the one that occurs more frequently. The various possibilities of improving the ACMS without having the user constantly collect data is the strength of ActiServ.

6.2.4 Summary and Discussion of Results

Since the Activity Recognition Service (ActiServ) consists of many steps, the results are now summarized and discussed. The recognition accuracy for the offline example implementation compared to the upper and lower baseline are shown in table 45. The numbers show, that reasonable recognition rates (71% for *conditional contexts*)

Service Step	Accuracy of Activity Recognition	Accuracy of Cond. Context Recognition	Delay Until Available
Upper Limit	97.3%	98.3%	days
Lower Limit	60.0%	78.9%	none
Selected ACMS	62.4%	71.2%	seconds
Pers. ACMS	86.0%	89.8%	hours
Pers. ACMS tested with diff. orientation	63.6%	90.6%	hours

Table 45: Recognition accuracy results.

and high rates ($> 90\%$) for some classes using initial data with ActiServ can be reached. This activity recognition is delivered through the service architecture with a delay of only seconds. The personalization step requires a longer computational period (a few hours), where the key factors for duration are the complexity of the ACMS and the efficiency of implementation of the PTS. After personalization recognition rates of up to 90% can be reached, with the exception of two classes, where this low accuracy could be accounted for through a better tradeoff in the PTS algorithms. With the ActiServ system also the lower baseline limit for opposite phone orientations not included in the GTS training (11.7 *PP* for *conditional contexts*) can be exceeded.

In general, with the GTS constantly training new combinations of ACMS on the user data in the database, the upper limit of over 97% accuracy can be reached with a runtime duration of a few days. This duration strongly depends on the server performance, the amount of user data in the database and the strategy with which the user data is selected for training. A random selection is the easiest choice of combining data to provide fast initial evaluation results, but in practice, a more sofisticated approach will be used. First ideas about the parameter based combination of user data were described in this section.

6.3 Summary and Conclusion

In this section an Activity Recognition Service (ActiServ) was presented, which aims to support activity recognition on mobile phones for common users. No user knowledge of machine learning is needed, nor are behavioral changes required in order to achieve results. Also, the effort of the user to collect data is kept to a minimum of 2-3 minutes per activity. Due to the combination of various users data, the resulting modular activity recognition is able to classify correctly even if the phone is not exactly carried in the position in which it was when the user collected the data.

The service provides an interface for annotating initial data, which then suffices for selection of the proper classification modules. Back-end services constantly improve recognition through personalization and simultaneously optimize recognition throughout the ActiServ user community. An evaluation for each of the service's main thread was presented and compared the results to upper and lower limits. The results indicate that ActiServ produces recognition rates of over 97% for the individual user, while also improving results for all other users of the service.

Another huge advantage of ActiServ is its capability to support the extension, adaption and exchange of the activity recognition. The general possibilities of the modular activity recognition towards the challenges **flexibility**, **extensibility**, **robustness**, **resources** and **conditionality** have been analyzed and experimentally evaluated in section 4. The ActiServ makes these capabilities accessible for the common user without the user having any knowledge about machine learning, data collection or activity recognition. The ActiServ combines the strength of the modular activity recognition with the strengths of a community based sensing approach to support the common user not the researcher. ActiServ deals with each of the challenges accordingly:

- **Flexibility:** ActiServ can adapt or exchange certain modules of a users running activity recognition if they are faulty or badly performing. The user can trigger this process through the Accuracy Feedback Interface (AFI). The adaption or exchange might demand the recollection of sensor data by the user with the Data Collector Tool (DCT). The adaption of classifier modules only involves the Personal Trainer Service (PTS), where the exchange both service parts PTS and Global Trainer Service (GTS) are needed.
- **Extensibility:** ActiServ can **extend** a classifier queue by additional modules. Therefore, the user has to provide sensor data over the DCT of the new activities to be recognized. For the **extension** of the modular activity recognition both PTS and GTS are needed.
- **Robustness & Resources:** ActiServ can also control the threshold for the filtering and the parameters of the sleep-time scheduling. Other mechanisms of the module scheduling or the filtering can be influenced or changed due to ActiServ if better methods are found. Therefore, ActiServ can be used to update the control of the activity recognition system.
- **Conditionality:** ActiServ can react on different **conditionalities** of the user by providing suitable modules to cope with this **conditions**. Therefore, the user has to provide data over the DCT which describe these **conditions** or the user has to give detailed feedback about the **condition**.

As can be seen, ActiServ is a very strong method to support activity recognition on common users mobile phones. It is especially suited to handle the novel modular activity recognition, but a service which offers **flexibility**, **extensibility**, **robustness**, **resource** friendliness and **conditionality** awareness would not be possible without the modular activity recognition proposed in this thesis.

7 Conclusions

In this thesis a novel modular activity recognition for mobile phones was presented. Although the recognition with accelerometer sensors on mobile phones is very challenging, a system was presented which can cope with these challenges. The architecture of the modular activity recognition was described in detail and evaluated according to the previously defined five challenges. Additionally a service to support the activity recognition for use on common users mobile phones was presented.

During the evaluation a surprising result was discovered. As it turned out the modular approach is not only better than a monolithic classifier, but the classification with only one class per module is by far the best in calculation effort and is among the highest in accuracy. It seems that the overhead for the complementary class is not rising according to the amount of modules. Although, due to the other improvements to the activity recognition instead of the modularity, the monolithic approach is not totally unsuitable for classification.

Some parts of the novel modular activity recognition and their evaluation have been presented in former publications, but the analysis and evaluation of the challenges **flexibility** and **conditionality** were totally unpublished so far. Some main parts of the evaluation to the challenges **robustness** and **resources** also have first been conducted for this thesis. Further investigations according to these challenges would be possible, but due to the extensiveness of this thesis as it is, they were not considered.

7.1 Summary

This thesis started with the motivation of activity recognition on mobile phones. Two distinctions were made for the application, the *calm mobile phone* and all *classical* applications to activity recognition. The *calm mobile phone* was defined to be unobtrusive and pushed in the background until the user explicitly wants to interact with it. Therefore embarrassing, disturbing or dangerous situations can be prevented due to activity recognition. Since the mobile phone is mostly with the user and provides a strong computation and sensing platform, *classical* applications of activity recognition can be projected onto mobile phones. Furthermore, the challenges of activity recognition were defined, which were refined after many years of experience with activity recognition on mobile phones. Along these challenges the proposed modular activity recognition was explained, analyzed and evaluated. All the contributions to activity recognition which help coping with each challenge were listed. At the end of the first section, all the previous publications including content of this theses were listed.

In the second section the basic principles of activity recognition and all the related work was listed. First, all the systems for activity recognition according to their application field were described briefly, where at the end a table summarizing all the work that is more or or less comparable to the proposed modular activity recognition was shown. Second, the basics of sensor data processing was explained along with the systems which implement the respective method. The chosen methods for activity recognition in this thesis were outlined. Last, the different methods for classifier fusion were listed also along systems, preferably for activity recognition, which implement them. The differences towards the modular approach presented in this thesis were described.

The third section was all about the architecture of the novel modular activity recognition. First, the data processing queue was defined, starting with the sensor and ending with a filtered tuple of class identifier and reliability measure. Additionally the bit-vector masking was defined. Second, the modularity of the classification process was specified. Various methods of arranging the modules were explained, but the only one used throughout this thesis was the dynamic queue concept. Third, the novel machine learning algorithm was described, which is used to train the mapping functions of the classifier modules. The algorithm consists of two different clustering techniques, linear regression and a genetic algorithm. Additionally the identification of the bit-vector, which is used for masking the mapping functions, with a genetic algorithm was explained. Last, all the tools, formats and implementations to collect and annotate sensor data, to exchange data and classifier modules and to support three different mobile phone architectures were described.

In the fourth section the capabilities of the modular activity recognition to cope with the challenges of recognition on mobile phones were analyzed. For **flexibility** the adaption and exchange of classifier modules was analyzed. To make the classifier queue **extensible** not just modules need to be added, but also the previous modules be bit-vector masked so they recognize transitions onto the new ones. Third, the capabilities of the system to provide **robustness** was analyzed. Due to the recurrent mapping function and the reliability filtering the classification process is stabilized and highly accurate. The **resources** of mobile phones are limited, so the proposed system was

analyzed on computational effort and energy consumption. Fifth, the modularity was investigated on the capabilities to react onto different **conditionalities**. Each module can especially be trained to react onto another *conditional context*. Last, two tradeoffs were analyzed: one between accuracy and event detection rate, another one between the challenges **robustness** and **resources**.

The experimental evaluation of the modular activity recognition according to the five challenges was presented in the fifth section. All the evaluation is based on actual accelerometer data from mobile phones of different users. Most of the solutions to the challenges were evaluated offline, only for the challenge **resources** some measurements were made on a mobile phone. At the end the two exemplary tradeoffs were experimentally evaluated.

A service called ActiServ for the support of the modular activity recognition on mobile phones was defined in section six. ActiServ not only reduces the effort of the single user by combining the training data with data of other users, but also provides **flexibility**, **extensibility** and **conditionality**. Several components, work flows and service parts were defined. ActiServ is not the main contribution to the topic activity recognition on mobile phones, but it rounds up the solution and makes the proposed modular activity recognition applicable by the common user.

7.2 Contribution

In this thesis a novel modular activity recognition was presented, which is especially suited for mobile phones. For the activity recognition on mobile phones first five challenges were identified, on which the proposed system has been evaluated. The challenges are **flexibility**, **extensibility**, **robustness**, **resources** and **conditionalities**.

The main contribution to the activity recognition, the modularity of the classification process, delivers solutions for the challenges **flexibility**, **extensibility**, **resources** and **conditionality**. Since the conditions of the users always can change, the adaption and exchange of certain parts of the recognition is necessary from time to time to offer **flexibility**. Due to the modularity of the recognition, not the whole system needs to be adapted or exchanged, but only the respective modules affected due to the changes. For different users and applications the recognition of different activities is important, but all demands together can not be supported by one system. Therefore, the modularity of the classification process makes the activity recognition **extensible** to the individual needs. Due to the modularity not the whole classifier needs to be calculated for every classification, but mostly only a fraction. This saves processing **resources** and indirectly also battery capacity. The user and the user's phone are also situated in very different **conditions**, which need representation in the classification process, because they result in very different sensor data patterns. With the modular activity recognition each module can be responsible for one *conditional context*, which reduces the calculation effort and increases the accuracy compared to a monolithic classifier.

Another contribution to the activity recognition with mobile phones are the covariant membership functions used in the classification process. These functions are especially suited for highly correlated sensor data as multi-axis accelerometers provide. Therefore, the covariant membership functions increase the accuracy of the activity recognition and provide a more semantically correct feature vector mapping, which increases the expressiveness of the desired reliability measure. This measure is used in a filtering, which increases the **robustness** of the recognition.

Also a significant improvement to the activity recognition is the recurrence in the classification process. The feedback of the mapping functions not only stabilizes the recognition, but enables the derivation of the reliability measure. Both, the stabilization and the reliability measure ensure the **robustness** of the activity recognition on mobile phones.

Due to the recurrent mapping function, the covariant membership function and the modularity of the classification process usual machine learning algorithms could not be applied. Therefore, a novel composition of different clusterings, linear regression and genetic algorithm in a machine learning algorithm was developed. This training algorithm delivers the desired classifier modules in a very efficient way.

The last contribution to activity recognition which was presented in this thesis is the novel method for adaption, personalization and generalization of the classifier modules. The adaption of the classifier modules is needed to provide **flexibility** if certain conditions of the user or her device have changed. When new modules get added to a classifier queue, the old modules need to be slightly to be modified, so they recognize transitions on the new modules in their *complementary class*. This is to enable the **extensibility** of the modular activity recognition. The service supporting the activity recognition on the common user's mobile phone needs the personalization of a set of classifier modules on the user. All these is possible with the novel bit-vector masking, that changes the classifier modules temporarily without destroying their original classification capabilities. The service called ActiServ can

also be seen as an innovation, since it is used to support the novel modular activity recognition on the common user's mobile phone and providing **flexibility**, **extensibility**, **robustness**, **resources** and **conditionalities**.

7.3 Future Work

As the evaluation has shown, there is much more to investigate than presented in this thesis. Every aspect and solution for one of the five challenges could be analyzed in more detail. For the solution to the challenge **flexibility** more combinations of adaption and exchange of modules could be tested with various datasets. A general understanding of the capabilities of the modular activity recognition according to adaption and exchange could be derived. The **extensibility** of classifier queues also could be investigated further. Questions like *by how many modules can existing queues be extended* or *is there a general limit to this approach* could be answered in the future. The recurrent mapping function used to provide **robustness** could be extended by having more past time steps included in the mapping. Also, how can the reliability measure be used in a reasoning[1] process and how can the overall accuracy be improved by it.

For the challenge **resources** many more experiments could be conducted. An extensive test of the modular activity recognition with and without sleep-time scheduling on different mobile phones could indicate where the classification and the scheduling could further be improved to save calculation and energy **resources**. Since the presented sleep-time scheduling is very simple, other scheduling algorithms could be tested on how they influence the energy consumption and classification accuracy.

The least researched capabilities are for the challenge **conditionality**. There only a general understanding of the principle was derived from the evaluation presented in this thesis. Data sets with many more different *conditional contexts* could indicate where the limit of the modular activity recognition lies in dealing with this challenge. An experiment about the possibilities to deal with different orientations of the phone with similar **conditionalities** could be conducted.

All solutions together can be evaluated in an online experiments in conjunction with ActiServ. There the participation of a large set of users can give a general understanding of the service and the possibilities to react onto the different challenges. Also, no hard numbers about the parameters of ActiServ could be derived so far, since no online tests were conducted which could show at which point what kind of response by the service is needed. Especially the training of the classifier modules and the identification of the bit-vectors need strategies to optimize runtime and accuracy, which can only be estimated when ActiServ is provided to a large set of users.

Furthermore, the different algorithms for classification despite the used fuzzy inference system (FIS) will be investigated if they could be used instead or in conjunction. Therefore, certain extensions to the original algorithms need to be made. The suggested methods for scheduling the classifier modules instead of the dynamic queue concept have also not been evaluated so far. Especially the probabilistic overlay to the fuzzy modular classification could improve performance and accuracy further. This kind of probabilistic model, as mentioned before, needs large naturalistic data sets about the user behavior and the usual period length of each activity, which to collect has not been possible for this thesis.

Finally, the methods to speed up the training of the classifier modules and the bit-vector identification need to be researched. Earlier it was mentioned, that the bit-vector search could be speeded up if certain knowledge about the data overlapping can be utilized. Therefore, the transformation of certain alpha cuts of the Gaussian membership functions into multidimensional ellipsoids could deliver a general volume metric about function overlapping, but this is a research topic of its own and was out of focus of this thesis.

[1] Initial experiments have been presented in [23].

References

[1] http://www.scipy.org/PerformancePython.

[2] Enhanced learning for evolutive neural architecture (elena) database. ftp://ftp.dice.ucl.ac.be/pub/neural-nets/ELENA/database.

[3] GStreamer – open source multimedia framework.

[4] Wearit@work. http://www.wearitatwork.com/.

[5] Wikipedia: Receiver operator characteristic. http://en.wikipedia.org/wiki/Receiver_operating_characteristic.

[6] S. Abbate, M. Avvenuti, P. Corsini, J. Light, and A. Vecchio. *Wireless Sensor Networks: Application-Centric Design*, chapter Monitoring of Human Movements for Fall Detection and Activities Recognition in Elderly Care Using Wireless Sensor Network: a Survey. InTech, 2010.

[7] J. Abonyi, R. Babuska, and F. Szeifert. Modified gath-geva fuzzy clustering for identification of takagi-sugeno fuzzy models. *Systems, Man and Cybernetics (SMC'02)*, 32, 2002.

[8] O. A. AbuAarqob, N. T. Shawagfeh, and O. A. AbuGhneim. Functions defined on fuzzy real numbers according to zadeh's extension. *International Mathematical Forum*, 3(16):763 – 776, 2008.

[9] S.-A. Ahmadi, N. Padoy, S. M. Heining, H. Feussner, M. Daumer, and N. Navab. Introducing wearable accelerometers in the surgery room for activity detection. In *7. Jahrestagung der Deutschen Gesellschaft fuer Computer-und Roboter-Assistierte Chirurgie (CURAC 2008)*, Leipzig, Germany, September 2008.

[10] E. M. Albornoz, D. H. Milone, and H. L. Rufiner. Hierarchical classifiers approach for emotions recognition. In *XIII Reunion de Trabajo en Procesamiento de la Informacion y Control (RPIC 2009)*, pages 853–858, Rosario, Argentina, September 16-18 2009.

[11] R. Ali, L. Atallah, B. Lo, and G.-Z. Yang. Transitional activity recognition with manifold embedding. *Proceedings of the 2009 Sixth International Workshop on Wearable and Implantable Body Sensor Networks (BSN'09)*, 2009.

[12] H. Altinçay. On naive bayesian fusion of dependent classifiers. *In Pattern Recognition Letters*, 26:2463–2473, November 2005.

[13] A. Assareh and L. Volkert. Fuzzy rule base classifier fusion for protein mass spectra based ovarian cancer diagnosis. In *Computational Intelligence in Bioinformatics and Computational Biology, 2009. CIBCB '09. IEEE Symposium on*, pages 193 –199, 30 2009-april 2 2009.

[14] L. Atallah, B. Lo, R. King, and G.-Z. Yang. Sensor placement for activity detection using wearable accelerometers. *Wearable and Implantable Body Sensor Networks, International Workshop on*, 0:24–29, 2010.

[15] A. Avci, S. Bosch, M. Marin-Perianu, R. Marin-Perianu, and P. Havinga. Activity recognition using inertial sensing for healthcare, wellbeing and sports applications: A survey. In 2010, editor, *23th International Conference on Architecture of Computing Systems (ARCS), 1st CoSDEO - Workshop on Context-Systems Design, Evaluation and Optimisation*, ISBN 978-3-8007-3222-7, pages 167–176, Hannover, Germany, 22-23 Feb 2010. VDE Verlag.

[16] M. Bahrepour, N. Meratnia, Z. Taghikhaki, and P. J. M. Havinga. Sensor fusion-based activity recognition for parkinson patients. In *Sensor Fusion - Foundation and Applications*, 2011.

[17] D. Bannach, K. Kunze, J. Weppner, and P. Lukowicz. Integrated tool chain for recording and handling large, multimodal context recognition data sets. In *Proceedings of the 12th ACM international conference adjunct papers on Ubiquitous computing*, Ubicomp '10, pages 357–358. ACM, 2010.

[18] D. Bannach and P. Lukowicz. Integrated tool chain for recording, handling, and utilizing large, multimodal context data sets for context recognition systems. *2nd Workshop on Context-Systems Design, Evaluation and Optimisation (CoSDEO), in conjunction with the 24th International Conference on Architecture of Computing Systems (ARCS)*, 2011.

[19] O. Banos, M. Damas, H. Pomares, and I. Rojas. Automatic recognition of daily living activities based on a hierarchical classifier. In J. Cabestany, I. Rojas, and G. Joya, editors, *Advances in Computational Intelligence*, volume 6692 of *Lecture Notes in Computer Science*, pages 185–193. Springer Berlin / Heidelberg, 2011.

[20] L. Bao and S. S. Intille. Activity recognition from user-annotated acceleration data. *PERVASIVE*, 2004.

[21] M. Beigl. Memoclip: A location-based remembrance appliance. *Personal Ubiquitous Computing*, 4:230–233, January 2000.

[22] K. P. Bennett. *Combining support vector and mathematical programming methods for classification*, pages 307–326. MIT Press, Cambridge, MA, USA, 1999.

[23] M. Berchtold and M. Beigl. Increased robustness in context detection and reasoning using uncertainty measures - concept and application. *Ambient Intell. (AmI'09), LNCS*, 2009.

[24] M. Berchtold, M. Budde, and M. Beigl. Video: Activity recognition on mobile phones - why do we need it and how can it be done? Video on the 9th International Conference on Pervasive Computing (Pervasive'11), `http://www.youtube.com/watch?v=7QMH29_y6yw`, 2011.

[25] M. Berchtold, M. Budde, D. Gordon, H. Schmidtke, and M. Beigl. Actiserv: Activity recognition service for mobile phones. In *International Symposium on Wearable Computers (ISWC'10)*, pages 1 –8, oct. 2010.

[26] M. Berchtold, M. Budde, H. Schmidtke, and M. Beigl. An extensible modular recognition concept that makes activity recognition practical. *German Artificial Intelligence Conference (KI'10), LNAI*, 2010.

[27] M. Berchtold, C. Decker, T. Riedel, T. Zimmer, and M. Beigl. Using a context quality measure for improving smart appliances. *IWSAWC*, 2007.

[28] M. Berchtold, H. Günther, M. Budde, and M. Beigl. Scheduling for a modular activity recognition system to reduce energy consumption on smartphones. *2nd Workshop on Context-Systems Design, Evaluation and Optimisation (CoSDEO), in conjunction with the 24th International Conference on Architecture of Computing Systems (ARCS)*, 2011.

[29] M. Berchtold, T. Riedel, M. Beigl, and C. Decker. Awarepen - classfication probability and fuzziness in a context aware application. *Ubiquitous Intelligence and Computing (UIC), LNCS*, 2008.

[30] M. Berchtold, T. Riedel, C. Decker, M. Beigl, and C. Bittel. Quality of location: Estimation, system integration and application. *The 5th International Conference on Networked Sensing Systems (INSS 2008), IEEE*, June 17 - 19 2008.

[31] M. Berchtold, T. Riedel, K. van Laerhoven, and C. Decker. Gath-geva specification and genetic generalization of tsk fuzzy models. *Systems, Man and Cybernetics (SMC'08), IEEE*, 2008.

[32] A. J. S. Bernhard Schölkopf. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning)*. MIT Press, 2001.

[33] J. C. Bezdek. Pattern recognition with fuzzy objective function algorithms. *Plenum Press*, 1975.

[34] J. C. Bezdek and J. C. Dunn. Optimal fuzzy partitions: A heuristic for estimating the parameters in a mixture of normal distributions. *IEEE Transactions on Computers*, pages 835–838, 1981.

[35] G. Bieber, J. Voskamp, and B. Urban. Activity recognition for everyday life on mobile phones. In C. Stephanidis, editor, *Universal Access in Human-Computer Interaction. Intelligent and Ubiquitous Interaction Environments*, volume 5615 of *Lecture Notes in Computer Science*, pages 289–296. Springer Berlin / Heidelberg, 2009.

[36] T. Brezmes, J.-L. Gorricho, and J. Cotrina. Activity recognition from accelerometer data on a mobile phone. In *Proceedings of the 9'th International Work-Conference on Artificial Neural Networks (IWANN'09)*, pages 796–799. Springer, 2009.

[37] C. J. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

[38] A. Carroll and G. Heiser. An analysis of power consumption in a smartphone. In *Proceedings of the 2010 USENIX conference on USENIX annual technical conference*, USENIXATC'10, pages 21–21, Berkeley, CA, USA, 2010. USENIX Association.

[39] G. S. Chambers, S. Venkatesh, G. A. W. West, and H. H. Bui. Hierarchical recognition of intentional human gestures for sports video annotation. *16th International Conference on Pattern Recognition (ICPR'02)*, 2:1082–1085, 2002.

[40] D. Chen and H. werner Gellersen. Recognition and reasoning in an awareness support system for generation of storyboard-like views of recent activity. In *International Conference on Supporting Group Work*, pages 14–17. ACM Press, 1999.

[41] S. Chiu. Method and software for extracting fuzzy classification rules by subtractive clustering. *IEEE Control Systems Magazine, pp. 461-465*, 1996.

[42] S. Chiu. *Fuzzy Information Engineering: A Guided Tour of Applications*, chapter 9, Extracting Fuzzy Rules from Data for Function Approximation and Pattern Classification. John Wiley&Sons, 1997.

[43] Y. Cho, Y. Nam, Y.-J. Choi, and W.-D. Cho. Smartbuckle: Human activity recognition using a 3-axis accelerometer and a wearable camera. In *HealthNet '08*, pages 1–3. ACM, 2008.

[44] T. Choudhury, G. Borriello, and S. Consolvo, et al. The mobile sensing platform: An embedded activity recognition system. *IEEE Pervasive Computing*, 7(2):32–41, 2008.

[45] S. Consolvo and D. W. McDonald, et al. Activity sensing in the wild: a field trial of ubifit garden. In *CHI*. ACM, 2008.

[46] L. Cordella, I. Finizio, C. Mazzariello, and C. Sansone. Using behavior knowledge space and temporal information for detecting intrusions in computer networks. In *Pattern Recognition and Image Analysis*, volume 3687 of *Lecture Notes in Computer Science*, pages 94–102. Springer Berlin / Heidelberg, 2005.

[47] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

[48] I. Crk, F. Albinali, C. Gniady, and J. Hartman. Understanding energy consumption of sensor enabled applications on mobile phones. In *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, pages 6885 –6888, sept. 2009.

[49] L. Davis. *Genetic Algorithms and Simulated Annealing*. Morgan Kaufmann, 1987.

[50] M. de Sa, L. Carrico, and P. Antunes. Ubiquitous psychotherapy. *Pervasive Computing, IEEE*, 6(1):20 –27, jan.-march 2007.

[51] M. Eermes, J. Pärkka, J. Mantyjarvi, and I. Korhonen. Detection of daily activities and sports with wearable sensors in controlled and uncontrolled conditions. *IEEE Transactions on Information Technology in Biomedicine*, 12(1), 1. January 2008.

[52] A. Fábián, N. Győrbíró, and G. Hományi. Activity recognition system for mobile phones using the motionband device. In *Proceedings of the 1st international conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications*, MOBILWARE '08, pages 41:1–41:5. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2007.

[53] J. Favela, M. Tentori, L. A. Castro, V. M. Gonzalez, E. B. Moran, and A. I. Martínez-García. Activity recognition for context-aware hospital applications: issues and opportunities for the deployment of pervasive networks. *Mobile Networks and Applications*, 12:155–171, March 2007.

[54] T. Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861 – 874, 2006. ROC Analysis in Pattern Recognition.

[55] D. Figo, P. C. Diniz, D. R. Ferreira, and J. a. M. Cardoso. Preprocessing techniques for context recognition from accelerometer data. *Personal and Ubiquitous Computing*, 14:645–662, October 2010.

[56] K. Forster, P. Brem, D. Roggen, and G. Troster. Evolving discriminative features robust to sensor displacement for activity recognition in body area sensor networks. In *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2009 5th International Conference on*, pages 43 –48, dec. 2009.

[57] I. Gath and A. B. Geva. Unsupervised optimal fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 11(7), pp 773-781*, 1989.

[58] H. Ghasemzadeh, V. Loseu, E. Guenterberg, and R. Jafari. Sport training using body sensor networks: a statistical approach to measure wrist rotation for golf swing. In *Proceedings of the Fourth International Conference on Body Area Networks*, BodyNets '09, pages 2:1–2:8, 2009.

[59] G. Giacinto and F. Roli. Methods for dynamic classifier selection. In *Proceedings of the International Conference on Image Analysis and Processing*, pages 659 –664, 1999.

[60] H. Gjoreski, M. Gams, and I. Chorbev. 3-axial accelerometers activity recognition. In *ICT Innovations 2010 (Editor M. Gusev), Web proceedings*, pages 51–58, 2010.

[61] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.

[62] D. Gordon, H. Schmidtke, M. Beigl, and G. von Zengen. A novel micro-vibration sensor for activity recognition: Potential and limitations. In *Wearable Computers (ISWC), 2010 International Symposium on*, pages 1 –8, oct. 2010.

[63] H. Guenther, F. E. Simrany, M. Berchtold, and M. Beigl. A tool chain for a lightweight, robust and uncertainty-based context classification system (ccs). *23rd International Conference on Architecture of Computing Systems (ARCS'10), 1st Workshop on Context-Systems Design, Evaluation and Optimization (CoSDEO)*, February 2010.

[64] V. Gunes, U. D. L. Rochelle, M. Ménard, and P. Loonis. Fuzzy clustering with ambiguity for multi-classifiers fusion: Clustering-classification cooperation. In *EUSFLAT-ESTYLF Joint Conference*, pages 505–508, 1999.

[65] E. E. Gustafson and W. C. Kessel. Fuzzy clustering with a fuzzy covariance matrix. *IEEE Conference on Decision and Control including the 17th Symposium on Adaptive Processes (CDC'78), San Diego, California, pp 761-766*.

[66] N. Györbíró, A. Fábián, and G. Hományi. An activity recognition system for mobile phones. *Mobile Networks and Applications (MONET'08)*, 14(1):82–91, 2009.

[67] P. Hajek, L. Godo, and F. Esteva. Fuzzy logic and probability. In *In Uncertainty in Artificial Intelligence. Proc. of 11th conference*, pages 237–244, 1995.

[68] F. Hausdorf. Analysis of user behavior in data collection for supervised machine learning in activity recognition. Magisterarbeit, Technische Universität Braunschweig, Institut für Betriebsysteme und Rechnerverbund, 2011. Supervisor: Martin Berchtold.

[69] Z. He and L. Jin. Activity recognition from acceleration data based on discrete consine transform and svm. In *IEEE International Conference on Systems, Man and Cybernetics (SMC'09)*, pages 5041 –5044, oct. 2009.

[70] Z. He, Z. Liu, L. Jin, L.-X. Zhen, and J.-C. Huang. Weightlessness feature — a novel feature for single tri-axial accelerometer based activity recognition. In *19th International Conference on Pattern Recognition (ICPR 2008)*, pages 1 –4, dec. 2008.

[71] E. A. Heinz, K. Kunze, M. Gruber, D. Bannach, and P. Lukowicz. Using wearable sensors for real-time recognition tasks in games of martial arts – an initial experiment. *Proceedings of the 2nd IEEE Symposium on Computational Intelligence and Games (CIG)*, pages 98–102, 2006.

[72] A. Helal, E. Kim, and S. Hossain. Scalable approaches to activity recognition research. *Proceedings of the Workshop titled "How to do good activity recognition research? Experimental methodologies, evaluation metrics, and reproducibility issues," in conjunction with Pervasive*, May 17-20 2010. Helsinki, Finland.

[73] M. Helmi and S. AlModarresi. Human activity recognition using a fuzzy inference system. In *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'09)*, pages 1897 –1902, aug. 2009.

[74] Y.-J. Hong, I.-J. Kim, S. C. Ahn, and H.-G. Kim. Activity recognition using wearable sensors for elder care. In *Future Generation Communication and Networking, 2008. FGCN '08. Second International Conference on*, volume 2, pages 302 –305, 2008.

[75] Y.-J. Hong, I.-J. Kim, S. C. Ahn, and H.-G. Kim. Mobile health monitoring system based on activity recognition using accelerometer. *SIMPRA*, 18(4):446 – 455, 2010.

[76] F. Huenupán, N. B. Yoma, C. Molina, and C. Garretón. Confidence based multiple classifier fusion in speaker verification. *Pattern Recogn. Lett.*, 29:957–966, May 2008.

[77] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science and Engineering*, 9:90–95, 2007.

[78] P. Huuskonen, J. M, and V. K. Collaborative context recognition for mobile devices. In *Handbook of Ambient Intelligence and Smart Environments*, pages 257–280. Springer US, 2010.

[79] T. Huynh, M. Fritz, and B. Schiele. Discovery of activity patterns using topic models. In *Proceedings of the 10th international conference on Ubiquitous computing*, UbiComp '08, pages 10–19, New York, NY, USA, 2008. ACM.

[80] T. Huynh and B. Schiele. Analyzing features for activity recognition. In *Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies*, sOc-EUSAI '05, pages 159–163. ACM, 2005.

[81] J.-S. R. Jang. Anfis: Adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man and Cybernetics, 1993, vol. 23 pp. 665-685*, 1993.

[82] D. Karantonis, M. Narayanan, M. Mathie, N. Lovell, and B. Celler. Implementation of a real-time human movement classifier using a triaxial accelerometer for ambulatory monitoring. *Information Technology in Biomedicine, IEEE Transactions on*, 10(1):156 –167, 2006.

[83] H. A. Kautz. *A formal theory of plan recognition*. PhD thesis, Rochester, NY, USA, 1987. UMI Order No. GAX87-18947.

[84] T. Kenesei, B. Balasko, and J. Abonyi. A matlab toolbox and its web based variant for fuzzy cluster analysis. `http://www.mathworks.com/matlabcentral/fileexchange/7486-clustering-toolbox`, 2005.

[85] A. Khan, Y.-K. Lee, S. Lee, and T.-S. Kim. Human activity recognition via an accelerometer-enabled-smartphone using kernel discriminant analysis. In *5th International Conference on Future Information Technology (FutureTech),*, pages 1 –6, may 2010.

[86] A. Khan, Y.-K. Lee, S. Lee, and T.-S. Kim. A triaxial accelerometer-based physical-activity recognition via augmented-signal features and a hierarchical recognizer. *IEEE Transactions on Information Technology in Biomedicine*, 14(5):1166 –1172, sept. 2010.

[87] A. M. Khan, Y. K. Lee, and T.-S. Kim. Accelerometer signal-based human activity recognition using augmented autoregressive model coefficients and artificial neural nets. In *Engineering in Medicine and Biology Society (EMBS'08), 30th Annual International Conference of the IEEE*, pages 5172 –5175, aug. 2008.

[88] B. Kikhia, J. Bengtsson, K. Synnes, Z. Sani, and J. Hallberg. Creating digital life stories through activity recognition with image filtering. In *Aging Friendly Technology for Health and Independence*, volume 6159 of *Lecture Notes in Computer Science*, pages 203–210. Springer Berlin / Heidelberg, 2010.

[89] J. Kukkonen, E. Lagerspetz, P. Nurmi, and M. Andersson. Betelgeuse: A platform for gathering and processing situational data. *Pervasive Computing, IEEE*, 8(2):49–56, 2009.

[90] L. Kuncheva, J. Bezdek, and M. Sutton. On combining multiple classifiers by fuzzy templates. In *Fuzzy Information Processing Society - NAFIPS, 1998 Conference of the North American*, pages 193 –197, aug 1998.

[91] K. Kunze, P. Lukowicz, H. Junker, and G. Tr?ster. Where am i: Recognizing on-body positions of wearable sensors. *Location- and Context-Awareness (LOCA'05)*, pages 264–275, 2005.

[92] K. Kunze, P. Lukowicz, K. Partridge, and B. Begole. Which way am i facing: Inferring horizontal device orientation from an accelerometer signal. *IEEE Thirteenth International Symposium on Wearable Computers (ISWC'09)*, 2009.

[93] J. R. Kwapisz, G. M. Weiss, and S. A. Moore. Activity recognition using cell phone accelerometers. *5th International Workshop on Knowledge Discovery from Sensor Data (SensorKDD'11)*, 12:74–82, March 2011.

[94] J. Lester, T. Choudhury, G. Borriello, S. Consolvo, J. L, K. Everitt, and I. Smith. Sensing and modeling activities to support physical fitness. *Workshop paper in UbiComp '05 Workshop: Monitoring, Measuring and Motivating Exercise: Ubiquitous Computing to Support Physical Fitness*, 2005.

[95] J. Lester, T. Choudhury, N. Kern, G. Borriello, and B. Hannaford. A hybrid discriminative/generative approach for modeling human activities. In *In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI*, pages 766–772, 2005.

[96] M. Li, V. Rozgic´ and, G. Thatte, S. Lee, A. Emken, M. Annavaram, U. Mitra, D. Spruijt-Metz, and S. Narayanan. Multimodal physical activity recognition by fusing temporal and cepstral information. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 18(4):369 –380, aug. 2010.

[97] C.-F. Lin and S.-D. Wang. Fuzzy support vector machines. *IEEE Transactions on Neural Networks*, 13(2):464 –471, mar 2002.

[98] C. Lombriser, N. B. Bharatula, D. Roggen, and G. Tröster. On-body activity recognition in a dynamic sensor network. In *Proceedings of the ICST 2nd international conference on Body area networks*, BodyNets '07, pages 17:1–17:6. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2007.

[99] X. Long, B. Yin, and R. M. Aarts. Single-accelerometer-based daily physical activity classification. *31st Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 6107 – 6110, 3-6 September 2009.

[100] M. Luštrek and B. Kaluža. Fall detection and activity recognition with machine learning. *Informatica*, 33:205–212, 2009.

[101] D. Maguire and R. Frisby. Comparison of feature classification algorithm for activity recognition based on accelerometer and heart rate data. *9th. IT & T Conference*, 2009.

[102] M. Marin-Perianu, C. Lombriser, O. Amft, P. Havinga, and G. TrÃ¶ster. Distributed activity recognition with fuzzy-enabled wireless sensor networks. In *Distributed Computing in Sensor Systems*, volume 5067 of *Lecture Notes in Computer Science*, pages 296–313. Springer Berlin / Heidelberg, 2008.

[103] U. Maurer, A. Smailagic, D. P. Siewiorek, and M. Deisher. Activity recognition and monitoring using multiple sensors on different body positions. In *BSN '06*. IEEE, 2006.

[104] S. Mckeever, J. Ye, L. Coyle, C. Bleakley, and S. Dobson. Activity recognition using temporal evidence theory. *Journal of Ambient Intelligence and Smart Environments*, 2:253–269, August 2010.

[105] J. Meyer, B. Poppinga, and S. Boll. Wellness 2.0 - sharing personal health experiences. *Wellness Informatics Workshop in conjunction with CHI 2010*, 2010.

[106] E. Miluzzo, N. D. Lane, K. Fodor, R. Peterson, H. Lu, M. Musolesi, S. B. Eisenman, X. Zheng, and A. T. Campbell. Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application. In *SenSys '08*. ACM, 2008.

[107] R. Muscillo, S. Conforto, and T. D.-A. Maurizio Sschmid. A hierarchical classifier to monitor adl through dynamic programming on dual-axis accelerometer data. *3rd WSEAS International Conference on REMOTE SENSING*, 2007.

[108] P. Nurmi, M. Przybilski, G. Lindén, and P. Floréen. A framework for distributed activity recognition in ubiquitous systems. In *IC-AI*, pages 650–655, 2005.

[109] U. of Waikato. Documentation to the weka toolkit. `http://wekadocs.com/node/2`.

[110] G. Ogris, T. Stiefmeier, P. Lukowicz, and G. Troster. Using a complex multi-modal on-body sensor system for activity spotting. In *Proceedings of the 2008 12th IEEE International Symposium on Wearable Computers*, pages 55–62, Washington, DC, USA, 2008. IEEE Computer Society.

[111] L. Oliveira, G. Monteiro, P. Peixoto, and U. Nunes. Towards a robust vision-based obstacle perception with classifier fusion in cybercars. In *Computer Aided Systems Theory EUROCAST 2007*, volume 4739 of *Lecture Notes in Computer Science*, pages 1089–1096. Springer Berlin / Heidelberg, 2007.

[112] N. Padoy, D. Mateus, D. Weinland, M.-O. Berger, and N. Navab. Workflow monitoring based on 3d motion features. In *Proceedings of the International Conference on Computer Vision Workshops, IEEE Workshop on Video-oriented Object and Event Classification*, 2009.

[113] S. Pirttikangas, K. Fujinami, and T. Nakajima. Feature selection and activity recognition from wearable sensors. In H. Youn, M. Kim, and H. Morikawa, editors, *Ubiquitous Computing Systems*, volume 4239 of *Lecture Notes in Computer Science*, pages 516–527. Springer Berlin / Heidelberg, 2006.

[114] T. Ploetz, N. Hammerla, and P. Olivier. Feature learning for activity recognition in ubiquitous computing. In *International Joint Conference on Artificial Intelligence (IJCAI'11)*, 2011.

[115] K. K. Rachuri, M. Musolesi, C. Mascolo, P. J. Rentfrow, C. Longworth, and A. Aucinas. Emotionsense: a mobile phones based adaptive platform for experimental social psychology research. In *Proceedings of the 12th ACM international conference on Ubiquitous computing*, Ubicomp '10, pages 281–290, New York, NY, USA, 2010. ACM.

[116] S. Raudys and F. Roli. The behavior knowledge space fusion method: Analysis of generalization error and strategies for performance improvement. In *Multiple Classifier Systems*, volume 2709 of *Lecture Notes in Computer Science*, pages 160–160. Springer Berlin / Heidelberg, 2003.

[117] N. Ravi, N. Dandekar, P. Mysore, and M. L. Littman. Activity recognition from accelerometer data. In *Proceedings of the 17th conference on Innovative applications of artificial intelligence (IAAI'05)*. AAAI Press, 2005.

[118] A. Reiss, G. Hendeby, G. Bleser, and D. Stricker. Activity recognition using biomechanical model based pose estimation. In P. Lukowicz, K. Kunze, and G. Kortuem, editors, *Smart Sensing and Context*, volume 6446 of *Lecture Notes in Computer Science*, pages 42–55. Springer Berlin / Heidelberg, 2010.

[119] D. Roggen, N. Bharatula, M. Stäger, P. Lukowicz, and G. Tröster. From sensors to miniature networked sensor buttons. In *Proceedings of the 3rd International Conference on Networked Sensing Systems - INSS 2006*, pages 119–122, June 2006.

[120] D. Roggen, K. Förster, A. Calatroni, A. Bulling, and G. Tröster. On the issue of variability in labels and sensor con?gurations in activity recognition systems. In *Workshop at the 8th International Conference on Pervasive Computing (Pervasive 2010)*, 2010.

[121] D. Ruta and B. Gabrys. An overview of classifier fusion methods. *Computing and Information Systems*, 7(1):1–10, 2000.

[122] D. Sanchez, M. Tentori, and J. Favela. Hidden markov models for activity recognition in ambient intelligence environments. *Mexican International Conference on Computer Science*, 0:33–40, 2007.

[123] T. S. Saponas, J. Lester, and J. E. Froehlich, et al. ilearn on the iphone: Real-time human activity classification on commodity mobile phones. *CSE Technical Report*, 2008.

[124] K. Schindler, L. V. Gool, and B. de Gelder. Recognizing emotions expressed by body pose: A biologically inspired neural model. *Neural Networks*, 21(9):1238 – 1246, 2008.

[125] A. Schmidt, K. A. Aidoo, and A. Takaluoma, et al. Advanced interaction in context. In *Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing (HUC'99)*. LNCS, 1999.

[126] A. Schmidt, M. Beigl, and H. w. Gellersen. There is more to context than location. *Computers and Graphics*, 23:893–901, 1998.

[127] D. Siewiorek, A. Smailagic, and J. Furukawa, et al. Sensay: A context-aware mobile phone. In *International Symposium on Wearable Computers (ISWC'03)*. IEEE Computer Society, 2003.

[128] S. Song, J. Jang, and S. Park. A phone for human activity recognition using triaxial acceleration sensor. *16th International Conference on Computers in Education (ICCE'08)*, 2008.

[129] S.-k. Song, J. Jang, and S. Park. An efficient method for activity recognition of the elderly using tilt signals of tri-axial acceleration sensor. In *Smart Homes and Health Telematics*, volume 5120 of *Lecture Notes in Computer Science*, pages 99–104. Springer Berlin / Heidelberg, 2008.

[130] F. Souvannavong and B. Huet. Continuous behaviour knowledge space for semantic indexing of video content. In *9th International Conference on Information Fusion*, pages 1 –7, july 2006.

[131] A. Subramanya, A. Raj, J. Bilmes, and D. Fox. Hierarchical models for activity recognition. In *IEEE 8th Workshop on Multimedia Signal Processing*, pages 233 –237, oct. 2006.

[132] M. Sugeno and G. Kang. Structure identification of fuzzy model. *Fuzzy Sets and Systems*, 1988.

[133] L. Sun, D. Zhang, B. Li, B. Guo, and S. Li. Activity recognition on an accelerometer embedded mobile phone with varying positions and orientations. In Z. Yu, R. Liscano, G. Chen, D. Zhang, and X. Zhou, editors, *Ubiquitous Intelligence and Computing*, volume 6406 of *Lecture Notes in Computer Science*, pages 548–562. Springer Berlin / Heidelberg, 2010.

[134] D. Tacconi, O. Mayora, P. Lukowicz, B. Arnrich, C. Setz, G. Troster, and C. Haring. Activity and emotion recognition to support early diagnosis of psychiatric diseases. In *Second International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth 2008)*, pages 100 –102, 302008-feb.1 2008.

[135] D. Tacconi, O. Mayora, P. Lukowicz, B. Arnrich, G. Tröster, and C. Haring. On the feasibility of using activity recognition and context aware interaction to support early diagnosis of bipolar disorder. In *UbiWell 2007 (Ubicomp Workshop)*, Innsbruck, Austria, September 2007.

[136] T. Tagaki and M. Sugeno. Fuzzy identification of systems and its application to modelling and control. *Systems, Man and Cybernetics (SMC'85)*, 1985.

[137] E. M. Tapia, S. S. Intille, W. Haskell, K. Larson, J. Wright, A. King, and R. Friedman. Real-time recognition of physical activities and their intensities using wireless accelerometers and a heart monitor. In *Proceedings of the 11th International Symposium on Wearable Computers (ISWC)*, 2007.

[138] O. Terrades, E. Valveny, and S. Tabbone. Optimal classifier fusion in a non-bayesian probabilistic framework. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(9):1630 –1644, sept. 2009.

[139] W. M. und W. Pitts. A logical calculus of the ideas immanent in nervous activity, 1943.

[140] T. van Kasteren, G. Englebienne, and B. Kröse. An activity monitoring system for elderly care using generative and discriminative models. *Personal and Ubiquitous Computing*, 14:489–498, 2010.

[141] T. van Kasteren, A. Noulas, G. Englebienne, and B. Kröse. Accurate activity recognition in a home setting. In *Proceedings of the 10th international conference on Ubiquitous computing*, UbiComp '08, pages 1–9, New York, NY, USA, 2008. ACM.

[142] K. Van Laerhoven and E. Berlin. When else did this happen? efficient subsequence representation and matching for wearable activity data. In *International Symposium on Wearable Computers (ISWC'09)*, pages 101 –104, 2009.

[143] M. Vankipuram, K. Kahol, T. Cohen, and V. L. Patel. Toward automated workflow analysis and visualization in clinical environments. *Journal of Biomedical Informatics*, In Press, Corrected Proof, 2010.

[144] V. Vapnik. The nature of statistical learning theory. 1995.

[145] G. Veres, H. Grabner, L. Middleton, , and L. V. Gool. Automatic workflow monitoring in industrial environments. In *In Proceedings Asian Conference on Computer Vision (ACCV)*, 2010.

[146] L. X. Wang. *Adaptive Fuzzy Systems and Control*. Prentice-Hall, Englewood Cliffs, 1998.

[147] S. Wang, J. Yang, N. Chen, X. Chen, and Q. Zhang. Human activity recognition with user-free accelerometers in the sensor networks. In *Neural Networks and Brain, 2005. ICNN B '05. International Conference on*, volume 2, pages 1212 –1217, oct. 2005.

[148] R. Want, V. Falcao, and J. Gibbons. The active badge location system. *ACM Transactions on Information Systems*, 10:91–102, 1992.

[149] J. Ward, P. Lukowicz, G. Troster, and T. Starner. Activity recognition of assembly tasks using body-worn microphones and accelerometers. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(10):1553 –1567, oct. 2006.

[150] M. Weiser and J. S. Brown. Designing calm technology. *World Wide Web Internet And Web Information Systems*, 1:1–5, 1995.

[151] T. Westeyn, K. Vadas, X. Bian, T. Starner, and G. D. Abowd. Recognizing mimicked autistic self-stimulatory behaviors using hmms. *Wearable Computers, IEEE International Symposium*, 0:164–169, 2005.

[152] B. Xiao, C. Wang, and R. Dai. Adaptive combination of classifiers and its application to handwritten chinese character recognition. *Int. Conference on Pattern Recognition*, 2000.

[153] R. Yager and D. Filer. Generation of fuzzy rules by mountain clustering. *Journal on Intelligent Fuzzy Systems, vol 2, pp 209-219*, 1994.

[154] J. Yamato, J. Ohya, and K. Ishii. Recognizing human action in time-sequential images using hidden markov model. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'92)*, pages 379 –385, jun 1992.

[155] J.-Y. Yang, Y.-P. Chen, G.-Y. Lee, S.-N. Liou, and J.-S. Wang. Activity recognition using one triaxial accelerometer: A neuro-fuzzy classifier with feature reduction. In *ICEC*, pages 395–400. Springer, 2007.

[156] J.-Y. Yang, J.-S. Wang, and Y.-P. Chen. Using acceleration measurements for activity recognition: An effective learning algorithm for constructing neural classifiers. *Pattern Recognition Letters*, 29(16):2213 – 2220, 2008.

[157] S.-I. Yang and S.-B. Cho. Recognizing human activities from accelerometer and physiological sensors. In *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI'08)*, pages 100 –105, aug. 2008.

[158] L. A. Zadeh. Fuzzy sets. volume 8, pages 338–353. Information and Control, 1965.

[159] L. A. Zadeh. Discussion: Probability theory and fuzzy logic are complementary rather than competitive. *Technometrics*, 37:271–276, 1995.

[160] P. Zappi, C. Lombriser, T. Stiefmeier, E. Farella, D. Roggen, L. Benini, and G. Troester. Activity recognition from on-body sensors: Accuracy-power trade-off by dynamic sensor selection. *European Workshop on Sensor Networks (EWSN'08)*, 2008.

[161] P. Zappi, T. Stiefmeier, E. Farella, D. Roggen, L. Benini, and G. Troster. Activity recognition from on-body sensors by classifier fusion: sensor scalability and robustness. In *Intelligent Sensors, Sensor Networks and Information, 2007. ISSNIP 2007. 3rd International Conference on*, pages 281 –286, 2007.

[162] P. Zappi, T. Stiefmeier, E. Farella, D. Roggen, L. Benini, and G. Troster. Activity recognition from on-body sensors by classifier fusion: sensor scalability and robustness. In *3rd International Conference on Intelligent Sensors, Sensor Networks and Information (ISSNIP'07)*, pages 281 –286, dec. 2007.

[163] A. Zell. *Simulation Neuronaler Netze*. Oldenbourg, 1994.

[164] T. Zhang, J. Wang, P. Liu, and J. Hou. Fall detection by embedding an accelerometer in cellphone and using kfd algorithm. *International Journal of Computer Science and Network Security*, 6:277–284, 2006.

[165] T. Zhang, J. Wang, L. Xu, and P. Liu. Fall detection by wearable sensor and one-class svm algorithm. In D.-S. Huang, K. Li, and G. Irwin, editors, *Intelligent Computing in Signal Processing and Pattern Recognition*, volume 345 of *Lecture Notes in Control and Information Sciences*, pages 858–863. Springer Berlin / Heidelberg, 2006.

[166] X. Zhu, X. Wu, and Y. Yang. Dynamic classifier selection for effective mining from noisy data streams. In *Data Mining, 2004. ICDM '04. Fourth IEEE International Conference on*, pages 305 – 312, nov. 2004.